Using Histograms to Better Answer Queries to Probabilistic Logic Programs

Matthias Broecheler*

May 4, 2009

Abstract

Probabilistic logic programs (PLPs) define a set of probability distribution functions (PDFs) over the set of all Herbrand interpretations of the underlying logical language. When answering a query Q, a lower and upper bound on Q is obtained by optimizing (min and max) an objective function subject to a set of linear constraints whose solutions are the PDFs mentioned above. A common critique not only of PLPs but many probabilistic logics is that the difference between the upper bound and lower bound is large, thus often providing very little useful information in the query answer. In this paper, we provide a new method to answer probabilistic queries that tries to come up with a histogram that "maps" the probability that the objective function will have a value in a given interval, subject to the above linear constraints. This allows the system to return to the user a histogram where he can directly "see" what the most likely probability range for his query will be. We prove that computing these histograms is $\#P$-hard, and show that computing these histograms is closely related to polyhedral volume computation. We show how existing randomized algorithms for volume computation can be adapted to the computation of such histograms. A prototype experimental implementation is discussed.

1 Introduction

Since the introduction of quantitative logic programs by Shapiro [19], van Emden [20], and others, there has been extensive interest in logic programming with uncertainty. While these early frameworks were fuzzy in nature, Ng and Subrahmanian [16] introduced probabilistic logic programs by building on top of probabilistic logics studied earlier by several authors such as Hailperin [8], Fagin *et al.* [7], and Nilsson [18]. There has been much subsequent work in this vein [13, 14, 4].

A fundamental problem with all of these probabilistic logics is the assumption of *ignorance* — it is assumed that we do not know of any dependencies or correlations between the events represented in these logics. Given a probabilistic logic program Π

*The research presented in this paper was conducted jointly with Gerardo I. Simari under the supervision and guidance of Dr. V.S. Subrahmanian over a logical language $\mathcal{L}$, we write down an associated set $LC(\Pi)$ of linear constraints. Each (ground) rule in Π generates one constraint. In addition, we have one variable in $LC(\Pi)$ for each Herbrand interpretation for language $\mathcal{L}$. While the rules in Π constrain what interpretations satisfy Π, these variables denote the probability that a Herbrand interpretation I actually represents the true state of the world. Assuming that the Herbrand Base of $\mathcal{L}$ is denoted $B_{\mathcal{L}}$, this means the linear program has $2^{B_{\mathcal{L}}}$ variables in it, and $O(|grd(\Pi)|)$ constraints. In many of these cases, $2^{B_{\mathcal{L}}}$ is significantly larger than $|grd(\Pi)|$. A consequence of this — well known to those in the field — is that $LC(\Pi)$ is vastly underconstrained as the number of variables very often significantly exceeds the number of rules. This has profound implications for the prospective utility of probabilistic logics and probabilistic logic programs. When answering a query Q (think of a query for now as a logical formula), we need to find the "lower bound" probability low_Q such that every Herbrand interpretation satisfying Π also satisfies Q with probability greater than or equal to low_Q. Likewise, we want to find the "upper bound" probability up_Q such that every Herbrand interpretation satisfying Π also satisfies Q with probability less than or equal to up_Q. To find the tightest such interval $[low_Q, up_Q]$ of this type, we minimize and maximize (respectively) an objective function associated with Q. When the problem is underconstrained as in most cases, it is often the case that low_Q is very close to 0 and up_Q is very close to 1, providing the user who wants to know the probability of Q very little information about the true probability of Q. The example below shows a very simple probabilistic logic program.

Example 1 (Stock Example) *Consider a very simple probabilistic logic program* Π_{stock} *(using the syntax of [16]):*

r_1	$stim\_pkg$	$: [0.30, 0.90]$	$\leftarrow .$
r_2	$home\_sales\_up$	$: [0.25, 0.85]$	$\leftarrow .$
r_3	$up\_ibm \wedge up\_goog$	$: [0.40, 0.95]$	$\leftarrow home\_sales\_up : [0.65, 0.90].$
r_4	$up\_ibm \vee up\_goog$	$: [0.60, 0.95]$	$\leftarrow home\_sales\_up : [0.65, 0.85].$
r_5	up_ibm	$: [0.30, 0.80]$	$\leftarrow stim\_pkg : [0.70, 1.0].$

The first two rules intuitively say that there is a $30-90\%$ *probability that a stimulus package will be announced (today) and that there is a* $25-85\%$ *probability that there will be an economic report released (today) that home sales are up. Rule* r_3 *says that if such a home sales report is released today, then IBM and Google's stock price will go up tomorrow with* $40-95\%$ *probability. Rule* r_4 *says that when such a home sales report is released (today), there is a* $60-95\%$ *probability that either IBM or Google's stock price will be up tomorrow. The last rule says that if an economic stimulus package is announced today, then there is a* $30-80\%$ *probability that IBM's stock price will go up tomorrow.*[1] *Though this example is obviously very simplistic, the reader can easily see that probabilistic logic rules that state that certain stocks go up when certain conditions are true can easily be derived from historical stock market data. Clearly, a stock analyst would like to make decisions based on such data.*

[1]We don't introduce time in this paper for the sake of simplicity. But you can think of the propositional symbols in the heads of the last three rules intuitively denoting stock movements tomorrow, while all other propositional symbols in Π_{stock} refer to events today.

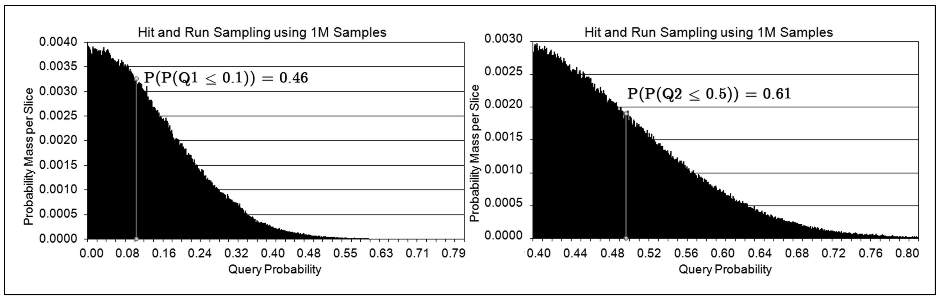

Figure 1: Histogram answers to queries Q_1 (left) and Q_2 (right) of Stock Example

According to the semantics of probabilistic logic programming [16], the probability of the conjunctive query $Q_1 = (stim\_pkg \wedge home\_sales\_up \wedge up\_ibm \wedge up\_goog)$ *is given by the interval* $[0, 0.8]$. *This is the tightest possible interval that we can infer for this query w.r.t.* Π_{stock}. *A stock analyst would have very little ability to "act" based on this answer, because the probability interval* $[0, 0.8]$ *is so wide that it basically tells the analyst very little. Past work in the AI community has often selected some value in this interval based on some principle (*e.g.*, maximum entropy, assuming independence, etc.). Worse still, the query* $Q_2 = up\_ibm$ *is entailed by* Π_{stock} *with tightest probability interval* $[0.4, 0.8]$. *Without any further information, the stock analyst may think that the probability of IBM going up is greater than 0.5, which might induce him to bet on IBM stock. However, the true story is that the probability of the probability of* up_ibm *being in the* $[0.4, 0.5]$ *interval is actually 61%.*

Moreover, no stock market analyst is going to want to risk millions of dollars of a mutual fund's investment based on what a probabilistic logic expert tells him (especially when that probabilistic logic expert knows nothing about the stock market and speaks in generalities about using maximal entropy, independence assumptions, etc.). The stock analyst wants to make these decisions, not rely on AI experts who do not understand the stock domain as well as he does.

Figure 1 shows visualizations of the histograms that we can present to such a stock analyst without making any additional assumptions *about the dependencies, correlations, etc. that may or may not exist, that the analyst may or may not believe, etc. The visualization shows a histogram for each query. The* x*-axis in Figure 1 (left), which corresponds to query* Q_1, *ranges from* 0 *to* 0.8 *(corresponding to the* $[0, 0.8]$ *interval associated with query Q). For a given point* x *in this interval, the histogram shows the probability that the probability of* Q *is at most* x. *Figure 1 (left) shows a sample value* x_0 *and its corresponding value* $h(x_0)$. *The histogram in Figure 1 (right) is similar and corresponds to query* Q_2.

The stock analyst has an immediate sense, by looking at the histogram in Figure 1 (right) that he should not bet on IBM stock going up. Likewise, the probability of query Q_1 *having probability* 0.5 *or more is low. However, there is no way for him to see this if we merely present him the interval* $[0.4, 0.8]$ *as the answer to the query. The histogram presents this interval (as the* x*-axis bounds), but it also shows far more valuable information that can enable the stock analyst to make a decision.*

***The goal of this paper is to show how to present answers of the kind mentioned above to the user so that we (i) present more information to the user than we did before, and (ii) so that this answer is expressed in an easy to understand graphical manner*.** We do this by using higher order probabilities.

The rest of this paper is organized as follows. In Section 2, we overview past work on PLPs from [16]. Then, in Section 3, we present the basic declarative semantics underlying histogram answers to PLP queries, and show that the histogram answer computation (HAC) for PLP queries is closely related to the problem of computing volumes of convex polytopes. In Section 4 we show that the HAC problem is $\#P$-hard; we also present two approximation algorithms for the HAC problem and prove appropriate complexity theorems. Section 5 contains implementation and experimental results showing that one of the approximation algorithms is far superior to the other.

2 Preliminaries

We now review (a simplified version of) the syntax and semantics of PLPs given in [17, 16]; there is nothing particularly new in this section.

2.1 Syntax of PLPs

We assume the existence of a set of propositional $\mathcal{L}_{pred}$ logic symbols. Every propositional symbol is an *atom*. Formulas are defined as follows.

Definition 1 (Formula) *An atom is a formula. If F_1 and F_2 are formulas, then $F_1 \wedge F_2$, $F_1 \vee F_2$, and $\neg F_1$ are formulas. Let $Form(\mathcal{L}_{pred})$ denote the set of all formulas.*

If F is a formula and $[\ell, u]$ is a subset of the real unit interval, $F : [\ell, u]$ is called an annotated *formula.*

Returning to Example 1, we can see that $stim\_pkg : [0.3, 0.9]$, $(up\_ibm \vee up\_goog) : [0.4, 0.95]$ and $(up\_ibm \vee up\_goog) : [0.65, 0.85]$ are annotated formulas. We now define the concept of *probabilistic rule*.

Definition 2 *If $F : \mu$, $B_1 : \mu_1, \ldots, B_m : \mu_m$ are annotated formulas, then $F : \mu \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_m : \mu_m$ is a* probabilistic rule. *If this rule is named r, then $Head(r)$ denotes $F : \mu$, and $Body(r)$ denotes $B_1 : \mu_1 \wedge \ldots \wedge B_m : \mu_m$.*

Intuitively, a probabilistic rule is a statement saying that if the formulas in the body are true with their associated probabilities, then the formula in the head is also true with its associated probability.

Definition 3 *A* probabilistic logic program *(*PLP*) is a finite set of probabilistic rules.*

Again, it is easy to see that in Example 1, Π_{stock} is a PLP. [2]

[2]We briefly note that the syntax presented here is — for the sake of space constraints — simpler than that in [16]. In particular, variable annotations, and function symbols over the annotation domain are eliminated. Moreover, [16] also removes the assumption of propositional logic and allows predicate symbols and first order logic atoms. However, the current framework can be easily extended to those cases. The definition of PLP above, however, does allow more complex formulas to appear both in the head and body of rules than the framework in [16]; in particular, negation (not a non-monotonic form of negation though) can appear in rule heads.

2.2 Semantics of PLPs

PLPs are characterized by a Krikpe style possible worlds semantics.

Definition 4 (World) *A* world *is any set of atoms.*

We use $\mathcal{W}$ to denote the set $2^{\mathcal{L}_{pred}}$ of all possible worlds. Since a world is simply a Herbrand interpretation, it is clear what it means for a world to *satisfy* a formula. A *probabilistic interpretation* is a probability distribution over worlds.

Definition 5 *Let S be a set of annotated formulas in $\mathcal{L}$, and $\mathcal{W}$ be the set of possible worlds. A* probabilistic interpretation *is a function $I : \mathcal{W} \to [0,1]$ such that $\sum_{w \in \mathcal{W}} I(w) = 1$.*

Definition 6 (Satisfaction) *Let $F : [\ell, u]$ be an annotated formula and I be a probabilistic interpretation. I is said to* satisfy *$F : [\ell, u]$ iff $\ell \leq \sum_{w_i \in \mathcal{W}, w_i \models F} I(w_i) \leq u$.*

Let $r = F : \mu \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_m : \mu_m$ be a probabilistic rule; I is said to satisfy *r iff either I satisfies Head(r) or I does not satisfy some $B_i : \mu_i \in$ Body(r).*

A probabilistic interpretation satisfies a PLP Π if and only if it satisfies all rules in Π. A PLP Π is said to be *consistent* if and only if there exists an interpretation I that satisfies all formulas in Π, and Π *entails* an annotated formula $F : \mu$ if and only if every interpretation that satisfies all rules in Π also satisfies $F : [\ell, u]$.

The above definition naturally leads to the definition of a system of linear constraints whose solutions will correspond to satisfying interpretations. We call this set *LC*(S), and it contains one variable p_i for each world $w_i \in \mathcal{W}$ and the following constraints:

1. For each $F : [\ell, u] \in S$, $\ell \leq \sum_{w_i \in \mathcal{W}, w_i \models F} p_i \leq u$, and
2. $\sum_{w_i \in \mathcal{W}} p_i = 1$

It follows immediately from [16], that S is consistent if and only if *LC*(S) is solvable.
Fixpoint Operator. Via a straightforward extension of a similar procedure in [17, 16], it is possible to associate a fixpoint operator T_Π with any PLP Π [3]. This operator maps sets of annotated formulas to sets of annotated formulas as follows and first involves defining an intermediate operator S_Π.

$$\begin{aligned} S_\Pi(X) \;\; = \;\; & \{F : \mu \mid (F : \mu \leftarrow B_1 : \mu_1 \wedge \cdots \wedge B_m : \mu_m) \in \Pi \wedge \\ & (\forall\, 1 \leq i \leq m)(\exists B'_i : \mu'_i \in X)(F_i = B_i \wedge \mu_i \subseteq \mu'_i)\}. \end{aligned}$$

For *each formula*[4] F, let $[\ell_F, u_F]$ denote the result of minimizing and maximizing $\sum_{w_i \in \mathcal{W}, w \models F} p_i$ subject to *LC*$(S_\Pi(X))$. We now define $T_\Pi(X)$ as follows.

$$T_\Pi(X) \;\; = \;\; \{F : [\ell_F, u_F] \mid F \in \mathit{Form}(\mathcal{L}_{pred})\}.$$

Using results similar to those in [17, 16], it is easy to show that T_Π has a least fixpoint and an annotated formula $F : [\ell, u]$ is a logical consequence of Π iff there is a formula $F : [\ell', u']$ in the least fixpoint such that $[\ell', u'] \subseteq [\ell, u]$.

[3]W.l.o.g., we assume that no rules in Π have formulas with a $[0, 1]$ annotation in the body.

[4]Many methods can be used to reduce the number of formulas in *Form*$(\mathcal{L}_{pred})$ we need to consider. Due to space constraints, and as this is not central to this paper, we ignore this issue.

3 Histogram Answers to a PLP Query

In classical PLPs, a query Q is an annotated formula $F : [\ell, u]$ and we want to check if Q is entailed by PLP Π (or alternatively if the least fixpoint of T_Π contains an annotated formula $F : [\ell', u']$ with $[\ell', u'] \subseteq [\ell, u]$). An alternative version says the query is a formula F and we want to find the annotated formula $F : [\ell', u']$ in the least fixpoint of T_Π. In this section, we propose a fundamentally different construct as the answer to the query that provides far more information to the user. Given a formula F as the query, we want to provide to the user a *histogram answer* to the query F w.r.t. a PLP Π. In order to do this, and in order to make our theory consistent with standard notation in (continuous) probability theory, we assume, without loss of generality, that all worlds in $\mathcal{W}$ are enumerated as $w_1, w_2, \dots, w_{|\mathcal{W}|}$ in some total order, and that an interpretation I is represented as a vector $(p_1, \dots, p_{|\mathcal{W}|})$ where each p_i denotes the probability of world w_i according to interpretation I, *i.e.*, $I(w_i) = p_i$. We now define the probability that a query formula will lie within a given probability interval.

Definition 7 (Higher-Order Probability of Entailment) *Suppose* Π *is a* PLP *and* Q *is a query formula. Suppose* $[a, b]$ *is a non-empty subset of* $[0, 1]$. *We define the* higher order probability that Q is entailed by Π with probability in $[a, b]$ *as:*

$$\mathbb{L}(a \leq Q \leq b \mid \Pi) = \int_{I \in \mathit{Mod}(\Pi)} \chi_{\left[a \leq \sum_{w_i \in \mathcal{W}, w_i \models Q} I(w_i) \leq b\right]} I d\mathbb{P}_\Pi I$$

where χ *is the adapted set membership function, i.e.,* $\chi_{C(x)} = 1$ *if* $C(x)$ *is true and* 0 *otherwise, for some condition C, and* $\mathbb{P}_\Pi$ *is the uniform probability distribution over* $\mathit{Mod}(\Pi)$*, the set of all interpretations that satisfy* Π*. Thus,* $\chi_{\left[a \leq \sum_{w_i \in \mathcal{W}, w_i \models Q} I(w_i) \leq b\right]} I$ *is true if interpretation* I *is such that* $\sum_{w_i \in \mathcal{W}, w_i \models Q} I(w_i)$ *lies between* a *and* b*.*

We now show that the expression above yields a valid probability distribution.

Theorem 1 *Given probability distribution* $\mathbb{P}_\Pi$ *over the set of interpretations that satisfy* Π*, a PLP* Π*, and some query formula Q,* $\mathbb{L}(Q = x \mid \Pi) = \mathbb{L}(x \leq Q \leq x \mid \Pi)$ *is a proper probability distribution over* $[0, 1]$.

Proof sketch. Let $\mathit{Mod}(\Pi)$ be the set of all interpretations that satisfy Π. Then:

$$\begin{aligned} &\textstyle\int_0^1 \mathbb{L}(Q = x \mid \Pi) \ = \ \mathbb{L}(0 \leq Q \leq 1 \mid \Pi) \\ &= \textstyle\int_{\mathit{Mod}(\Pi)} \chi_{\left[0 \leq \sum_{w_i \in \mathcal{W}', w_i \models Q} p_i \leq 1\right]} (I = (p_i)) d\mathbb{P}_\Pi I \\ &= \textstyle\int_{\mathit{Mod}(\Pi)} \chi_{[\text{true}]} (I = (p_i)) d\mathbb{P}_\Pi I = \int_{\mathit{Mod}(\Pi)} 1 d\mathbb{P}_\Pi I = 1 \end{aligned}$$

The last equality holds since $\mathbb{P}_\Pi$ is a probability distribution over $\mathit{Mod}(\Pi)$. □

We now return to Example 1 in order to illustrate the above definition of a higher order probability of entailment.

Example 2 *Consider the queries* Q_1 *and* Q_2 *of Example 1:*

- $\mathbb{L}(0 \leq Q_1 \leq 0.1 \mid \Pi_{\mathit{stock}})$. *This represents the probability that* Q_1 *is entailed by* Π_{stock} *with probability in the range* $[0, 0.1]$. *We compute this using Definition 7 by solving the integral:* $\int_{I \in \mathit{Mod}(\Pi)} \chi_{\left[0 \leq \sum_{w_i \in \mathcal{W}, w_i \models Q_1} I(w_i) \leq 0.1\right]} I d\mathbb{P}_\Pi I$.

- $\mathbb{L}(0.7 \leq Q_2 \leq 0.75 \mid \Pi_{stock})$. *This represents the probability that Q_2 is entailed by Π_{stock} with probability in the range $[0.7, 0.75]$. Similar to the first case, we compute this by solving:* $\int_{I \in Mod(\Pi)} \chi_{\left[0.7 \leq \sum_{w_i \in \mathcal{W}, w_i \models Q_2} I(w_i) \leq 0.75\right]} I d\mathbb{P}_{\Pi} I$.

Given a query formula Q, we can now ask for the probability that Q is entailed by PLP Π with point probability p or with a probability in the range $[a, b]$. The answer to these queries, respectively, are $\mathbb{L}(p \leq Q \leq p \mid \Pi)$ and $\mathbb{L}(a \leq Q \leq b \mid S)$. This gives us more information than simply knowing the widest interval $[\ell, u]$ of probability values for the entailment of Q. $\mathbb{L}$ gives us the entire distribution of probability values for a query formula and not just the smallest interval such that $\mathbb{L}(\ell \leq Q \leq u \mid \Pi) = 1$. *Thus, the higher order probability of entailment gives users strictly more information than answers in classical* PLP. Moreover, as shown in Figure 1, we can present the entire distribution of $\mathbb{L}$ for a given query Q, and enable a naive user (who has no in-depth knowledge of probability theory, and almost certainly no knowledge of higher order probabilities) to visualize the probability distribution for his query. There are two ways to do this. As Definition 7 provides a continuous probability distribution, we can just present an approximation of the continuous histogram as shown in Figure 1, or we can also present a *discrete* version of this answer.

Definition 8 (Histogram Answer) *Suppose Π is a PLP and Q is a query. The* histogram answer *to query Q w.r.t. PLP Π is the function $\mathbb{L}$.*

Further suppose that $k \geq 1$ is an integer and that the $[\ell, u]$ is the tightest interval such that $\Pi \models Q : [\ell, u]$. The k-discrete histogram answer *to query Q w.r.t.* PLP Π *is the set* $\{\mathbb{L}(\ell + (i-1) * \frac{u-\ell}{k} \leq Q \leq \ell + i * \frac{u-\ell}{k} \mid \Pi) \mid 1 \leq i \leq \frac{u-\ell}{k}\}$.

If the user wants a discrete (rather than a continuous) histogram answer, then he can select an integer k which specifies the desired level of discretization. The k-discrete histogram answer splits the tightest $[\ell, u]$ interval such that $\Pi \models Q : [\ell, u]$ into k equally sized sub-intervals. For each of these subintervals, it finds the probability that Q's probability lies in that sub-interval using the formula given above. The following theorem shows that computing these histograms is closely related to the problem of volume computation in convex polyhedra.

Theorem 2 *Let $\mathbb{P}_{\Pi}$ be the uniform probability distribution over $Mod(S)$, Q a query formula, and $[a, b] \subseteq [0, 1]$. Then:*

$$\mathbb{L}(a \leq Q \leq b \mid \Pi) = \frac{vol\left(SOL\left(a \leq \sum_{w_i \in \mathcal{W}, w_i \models \Pi, w_i \models Q} prob(w_i) \leq b\right)\right)}{vol\left(SOL(LC(\Pi))\right)}$$

where $SOL(X)$ denotes the set of solutions of a set of constraints X, and $vol(B)$ denotes the m dimensional volume of a set of points B that form an m dimensional body in Euclidean space[5] .

[5]Note that the solutions of $LC(\Pi)$ and the models of the PLP Π are in exact one to one correspondence, so we could speak interchangeably here about either solutions of $LC(\Pi)$ or models of Π. We prefer speaking about solutions of $LC(\Pi)$ as we are using geometric intuitions here in computing polytope volumes.

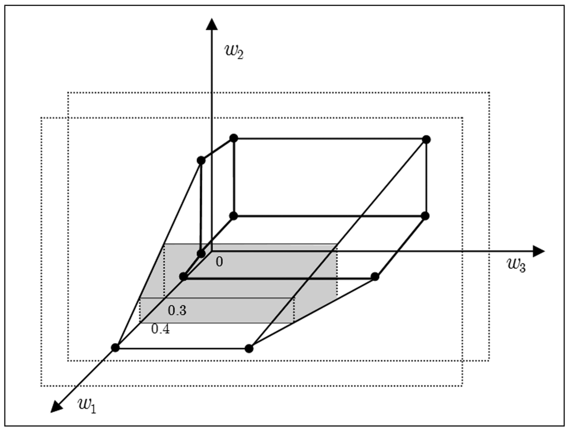

Figure 2: The polytope from Example 3 intersected by the two hyperplanes that are determined by the query formula and its probability interval (region corresponding to Q is shown shaded).

For ease of notation, we will denote the numerator of the above expression by $\mathit{Mod}(\Pi)(a \leq Q \leq b) = \Big\{ I \in \mathit{Mod}(\Pi) \mid a \leq \sum_{w_i \in \mathcal{W}, w_i \models \Pi, w_i \models Q} I(w_i) \leq b \Big\}$.
Proof sketch. $\mathbb{L}(a \leq Q \leq b \mid \Pi) =$

$$= \frac{\int_{I \in \mathit{Mod}(\Pi)} \chi_{\left[a \leq \sum_{w_i \in \mathcal{W}, w_i \models Q} I(w_i) \leq b\right]} I d\mathbb{P}_\Pi I}{1}$$

$$= \frac{\int_{I \in \mathit{Mod}(\Pi)} \chi_{\left[a \leq \sum_{w_i \in \mathcal{W}, w_i \models Q} I(w_i) \leq b\right]} I d\mathbb{P}_\Pi I}{\int_{\mathit{Mod}(\Pi)} 1 d\mathbb{P}_\Pi I}$$

$$= \frac{\mathit{vol}\left(\mathit{SOL}\left(a \leq \sum_{w_i \in \mathcal{W}, w_i \models \Pi, w_i \models Q} \mathit{prob}(w_i) \leq b\right)\right)}{\mathit{vol}\left(\mathit{SOL}(\mathit{LC}(\Pi))\right)}$$

□

Theorem 2 shows that computing the probability distribution $\mathbb{L}$ is closely related to volume computations on the convex polytope formed by the linear constraints in $\mathit{LC}(S)$ in n dimensional Euclidean space.

Example 3 *Suppose we have* $\Pi = \{a : [0.6, 0.9], b : [0.2, 0.5]\}$*, and the query formula is* $Q = a \wedge \neg b$*. The set of possible worlds is given by* $w_0 = \{\}$*,* $w_1 = \{a\}$*,* $w_2 = \{b\}$*, and* $w_3 = \{a, b\}$*. In the following, let* p_i *denote the probability of world* w_i *being true;* $\mathit{LC}(\Pi)$ *is given by:*

$$\{0.6 \leq p_1 + p_3 \leq 0.9, \quad 0.2 \leq p_2 + p_3 \leq 0.5, \quad p_0 + p_1 + p_2 + p_3 = 1\}$$

In this case, the query formula is satisfied only by world w_1*. Maximizing and minimizing the value of variable* p_1 *in the LP above yields as a result that* Q *is entailed with a probability in the interval* $[0, 0.5]$*. Figure 2 shows the geometric interpretation* *of these constraints. In the figure, we can see that if we are interested in knowing the probability that Q will be true with a probability between 0.3 and 0.4, then the region of interest is the one shown shaded.*

4 Volume Computation and Answer Histograms

As shown in the preceding section, computing $\mathbb{L}(a \leq Q \leq b \mid \Pi)$ can be reduced to the problem of computing the ratio between the two volumes $\{I \mid I \models \Pi \wedge a \leq I(Q) \leq b\}$ and *Mod*(Π). Compared to *Mod*(Π), $\{I \mid I \models \Pi \wedge a \leq I(Q) \leq b\}$ is also a convex polytope which is defined via the set of linear constraints *LC*(Π) and two additional constraints:

$$(1) \sum_{w_i \in \mathcal{W}, w_i \models Q} p_i \geq a, \quad (2) \sum_{w_i \in \mathcal{W}, w_i \models Q} p_i \leq b$$

We use *LC*(Π, Q, a, b) to refer to this modified set of constraints for a query Q.

Hence, we can build upon previous work on computing volumes of convex polytopes. A simple algorithm for the *discrete histogram answer* to PLP queries would work as follows and uses a function called vol that takes a set of linear constraints as input and returns the volume of the convex polytope generated by those linear constraints.

```
Algorithm DiscreteHistoAnswer$(\Pi, Q, k)$
1. Result $= \emptyset$;
2. Minimize and maximize $\sum_{w_i \in \mathcal{W}, w_i \models Q} p_i$ subject to $LC(\Pi)$ to get $\ell, u$ respectively;
3. Let $c = (u - \ell)/k$;
4. for $i = 1$ to $c$ do
    a. $V_{\ell+(i-1)*c, \ell+i*c} = \frac{vol(LC(\Pi, Q, \ell+(i-1)*c, \ell+i*c))}{vol(LC(\Pi, Q, \ell, u))}$;
    b. Add $V_{\ell+(i-1)*c, \ell+i*c}$ to $Result$;
5. return Result;
```

The following result states that this algorithm correctly computes the discrete histogram answer and follows immediately from Theorem 2.

Theorem 3 *Algorithm* ***DiscreteHistoAnswer***(Π, Q, k) *correctly computes the k-discrete histogram answer to this query.*

As the correctness of the above algorithm depends on volume computation algorithms, we provide a brief overview of those algorithms below. Cohen and Hickey [3] were the first to propose exact algorithms based on triangulation with exponential run time complexity, followed by Khachiyan [9] a decade later. Later, Dyer and Frieze [6] proved that computing the volume of a convex polytope defined by a set of constraints is $\#P$-hard, thereby showing that this is the best time complexity one can achieve for exact algorithms. Dyer *et al.* [5] proposed a randomized algorithm to compute arbitrarily tight bounds on the volume of convex polytopes with high probability in polynomial time. [12] presented an $O^*(n^4)$ randomized polynomial time (approximation) algorithm, where n is the dimensionality of the polytope[6].

[6]The O^* notation ignores logarithmic factors and other factors such as error bounds.

Due to the high dimensionality of the PLP histogram answer computation problem, exact volume computation algorithms are not going to work in practice. [2] study such algorithms and only consider cases with dimensionality below 20. Even in our very small stock market example, which has just 4 propositional symbols, we already have a 16-dimensional space as there are 16 possible worlds to consider!

The randomized volume computation algorithms use random walks with rapid mixing time[7] inside the polytope. Such random walks generate a Markov chain where each point in the polytope corresponds to a state in the Markov chain, and the transition probabilities denote the probability of the random walk taking you from one point to another. Sampling from this Markov Chain in accordance with the mixing time yields a uniform distribution over the polytope. Using this sampling strategy, one can compute the ratio between the volume of a known body (*e.g.*, the unit cube) and the polytope of interest. Naively applying existing volume computation algorithms to compute $\mathbb{L}(a \leq Q \leq b \mid \Pi)$ as given in the **DiscreteHistoAnswer** algorithm has two serious shortcomings:

1. We wish to plot a histogram of the distribution of $\mathbb{L}$, *i.e.*, for an interval width $\delta = \frac{u-l}{k}$. Computing each of the volumes $vol(LC(\Pi, Q, \ell+(i-1)*\delta, \ell+i*\delta))$ is expensive as the (already expensive) volume computation algorithm would need to be invoked $k+1$ times (once for each of the k discretized components, and once for the entire volume). This increases the running time by $O(k)$.

2. As stated before, computing $\mathbb{L}(a \leq Q \leq b \mid \Pi)$ requires the computation of the *ratio* between the two volumes and not the actual volume. This raises the question: can we somehow do better than volume computation algorithms?

The following theorem provides an answer to point (2).

Theorem 4 *Let K denote an arbitrary n dimensional polytope which is defined as the intersection of a set K_M of half-spaces. Let A, B be two additional half-spaces and let L denote the polytope which is the intersection of the half-spaces in $L_M = K_M \cup \{A, B\}$. Under these circumstances, computing $\frac{vol(L)}{vol(K)}$ is $\#P$-hard.*

Proof sketch. Dyer and Frieze [6] have proven that computing the volume of a convex polytope defined by the intersection of half-spaces is $\#P$-hard. We show how convex polytope volume computation can be reduced to relative volume computation in polynomial time, thereby establishing $\#P$-hardness of relative volume computation.

We assume that an arbitrary polytope K is defined by the intersection of a set of K_M of half-spaces. To compute the volume of K using relative volumes, we proceed as follows.

Firstly, we make the customary assumption that the origin o is inside K. We can determine the maximal inscribed n dimensional sphere inside K in time polynomial in the number of bounding half-spaces $|K_M|$. Let r be the radius of this maximal sphere, then we can fit a cube C of edge length $\ell = \frac{2r}{\sqrt{n}}$ centered at the origin inside this circle and hence C must be contained in K. For more details on how a contained cube can

[7]The term *mixing time* refers to the number of steps the random walk must take in order to reach its stationary distribution; see [15] for a complete treatment.

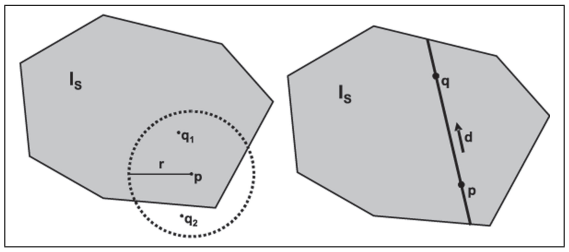

Figure 3: Schematic Ball Walk (left) and Hit-and-Run (Right)

be determined in polynomial time, the interested reader is referred to Applegate and Kannan [1] who proved that one can find an affine mapping in polynomial time which maps K to K' such that the unit cube is contained in K'.

We can compute the volume of C in closed form as $vol(C) = \ell^n$. Using this base volume we can derive the volume of K as follows. Let $\{F_j^i\}$ for $i = 1, \ldots, n$ and $j = 0, 1$ denote the set of faces of the cube C where F_0^i, F_1^i are parallel and opposing faces, for all i. For our purposes, we consider the faces to be half-spaces which bound the cube. Then C can be considered as the intersection of the n pairs of parallel half-spaces F_0^i, F_1^i. Now, let K_d denote the polytope defined as the intersection of half-spaces $K_M \cup \{F_0^i, F_1^i \mid i = 1, \ldots, d\}$ for $0 \leq d \leq n$. Then $K_0 = K$ and $K_n = C$, since C is contained in K. We can now derive the volume of K, $vol(K) =$

$$vol(K_0) = vol(K_1)\frac{vol(K_0)}{vol(K_1)} = vol(K_n)\prod_{d=1}^{n}\frac{vol(K_{d-1})}{vol(K_d)} = \ell^n \prod_{d=1}^{n}\frac{vol(K_{d-1})}{vol(K_d)}$$

Hence, we have reduced computing the exact volume of K to the product of n relative volume computations, which completes the polynomial reduction. □

4.1 The Approx-HOPE Algorithm

We now present the Approx-HOPE algorithm (short for the Approximate Histogram Oriented Probabilistic Entailment algorithm) which uses randomized methods to compute the histogram answer to a query Q w.r.t. a PLP Π. The Approx-HOPE algorithm uses a function called *randomWalk* that takes *LC*(Π) as input and performs a random walk through the convex polytope defined by Π. This function can be implemented in many ways, two of which we will discuss later.

```
**Algorithm** Approx-HOPE$(\Pi, Q, k)$
1. Result $= \emptyset$;
2. Let $\delta = (u - l)/k$;
3. Sample $= randomWalk(LC(\Pi))$;
4. **For** $i = 1$ **to** $\delta$ **do**
        a. $V_{\ell+(i-1)*\delta,\ell+i*\delta} = \frac{|Sample \cap [\ell+(i-1)*\delta,\ell+i*\delta]|}{|textitSample|}$;
        b. Add $V_{\ell+(i-1)*\delta,\ell+i*\delta}$ to Result;
5. **return** Result;
```

The Approx-HOPE algorithm is quite simple. Rather than solve volume computation problems $k+1$ times as the **DiscreteHistoAnswer** algorithm does, this algorithm basically executes one pass of the sampling stage of these randomized volume computation algorithms. All these algorithms sample from a polytope with a view to inferring the volume of the polytope. Rather than sample to determine the volume of the polytope, we try to use the random walk to estimate the part of the polytope's volume that lies within one of the k probability intervals that we are discretizing our problem into.

Though Approx-HOPE can be used with any appropriately designed random walk algorithm, we have tested it extensively with two well known ones:
(1) The **random ball walk (RBW)** starts at an arbitrarily chosen point $p \in SOL(\mathit{LC}(\Pi))$ where $SOL(X)$ denotes the set of solutions of a set X of constraints. It has a fixed associated parameter r which denotes the radius of a "ball" used during the random walk. To move to the next point, we uniformly sample a point q from the n dimensional sphere of radius r with center p. If q lies inside the polytope $\mathit{LC}(\Pi)$, the random walk moves to point q, otherwise the point is rejected and the walk stays at p. The procedure is then repeated at the selected (new or old) point. Figure 3 (left) visualizes the random ball walk and shows the point q_1 which would be accepted as the next move and q_2 which would be rejected.
(2) The **Hit-and-Run (HAR)** walk also starts at an arbitrary point $p \in SOL(\mathit{LC}(\Pi))$ and has no parameters. At each step, a direction d (*i.e.*, a point on the n dimensional sphere surface) is chosen uniformly at random. We compute the segment of line l inside the polytope $\mathit{Mod}(\Pi)$, where l is the line through p in direction d. Finally, a point q is chosen uniformly at random from this line segment and the walk moves to q. Figure 3 (right) shows a line segment inside the polytope and the next point q. Note that the Hit-and-Run walk *never rejects any points*.

It has been shown that both RBW and HAR have a mixing time of $O^*(n^3)$; however, HAR achieves this mixing time under weaker assumptions [11]. As we will see in Section 5, our experiments show that HAR performs much better in practice as it mixes much more rapidly. This is due to the fact that the random ball walk frequently gets "stuck" for large radii and moves only very slowly for small radii.

Theorem 5 *Using either the RBW or HAR sampling strategy, Approx-HOPE runs in time in $O^*(n^4 m)$, where m is the number of rules in Π and n is the number of worlds.*

Proof sketch. Sampling uniformly at random from a ball of radius r takes time linear in the number of dimensions n. Determining whether a point lies inside the polytope defined by $\mathit{LC}(\Pi)$, as required by RBW, as well as computing the line fragment for a given direction d, which is needed for HAR, can be done in time in $O(nm)$. □

5 Experiments

We implemented Approx-HOPE with both the RBW and HAR methods in Matlab 7.7.0 on a single machine with a 2.6 GHz Intel Core Duo Processor using only a single core and 3GB of RAM.

In our experiments, we randomly generated least fixpoints of PLPs. These fixpoints contained 3 to 10 annotated formulas, each with up to 4 propositional symbols

	500,000 Samples		1,000,000 Samples		2,000,000 Samples	
	Ball Walk	Hit-And-Run	Ball Walk	Hit-And-Run	Ball Walk	Hit-And-Run
3 rules, 7 worlds	13.7	23.1	28.6	46.7	56.1	91.4
4 rules, 15 worlds	14.6	23.8	29.7	47.7	58.3	95.6
5 rules, 31 worlds	15.4	26.1	30.7	52.2	62.1	102.6
6 rules, 59 worlds	16.5	29.7	32.9	60.4	65.3	119.0
7 rules, 71 worlds	17.0	31.5	34.0	63.3	68.3	127.5
8 rules, 112 worlds	19.9	38.4	40.5	76.4	76.7	153.5
9 rules, 159 worlds	25.9	46.4	52.0	93.4	100.7	180.8
10 rules, 239 worlds	38.5	65.2	77.1	130.7	149.6	259.8

Figure 4: Running times in seconds for varying numbers of worlds and rules.

in them. No fixpoint contained more than 12 propositional symbols in total. Though there should be 2^k worlds when there are k propositional symbols in such fixpoints, we eliminated some worlds using a world equivalence method described in [10], which is why the numbers of worlds in Figure 4 are not necessarily powers of two. We then recorded run times for the Approx-HOPE algorithm using the RBW and HAR sampling strategies and three different sample sizes. The running times in seconds are shown in Figure 4. As expected, the run times increase linearly with the number of samples for all rule sets. Moreover, the run time increases with the number of worlds, because the computational cost per sample depends on the number of worlds, as explained in the proof of Theorem 5. We observe that the RBW strategy outperforms the HAR strategy in running time since its cost per iteration is lower. Note that the sample sizes were identical for all rule sets, irrespective of the number of worlds and therefore irrespective of the mixing times.

In the qualitative experiments we studied the convergence of the RBW and HAR sampling strategies in detail by holding the rule set and query constant and varying the sample size between 100,000 and 40 million. Part of the results for a single experiment with 10 rules and 341 worlds are shown in Figure 5. Across all experiments we observed that HAR converges more quickly to the uniform distribution than RBW. As an example, Figure 5 shows that Approx-HOPE with HAR already clearly indicates the subinterval with the highest probability after only 1 million samples, whereas the RBW is still “walking” toward that region in the polytope. After 20 million samples, HAR has converged to the uniform distribution (*i.e.*, increasing the sample size does not change the histogram) whereas RBW is still far from convergence. We conclude that *the HAR sampling strategy significantly outperforms RBW*, despite its favorable cost per iteration, since HAR converges much more rapidly and requires significantly less iterations.

To verify the scalability of Approx-HOPE we experimented with a set of 15 rules giving rise to 682 worlds using different random queries. The HAR strategy converged to the uniform distribution after approximately 140 million samples, with a computation time of 10 hours.

6 Conclusion

Probabilistic logic programming has been studied for almost 25 years [8, 7, 18, 16, 13, 14, 4]. For most of these years, researchers have known that the probability in-

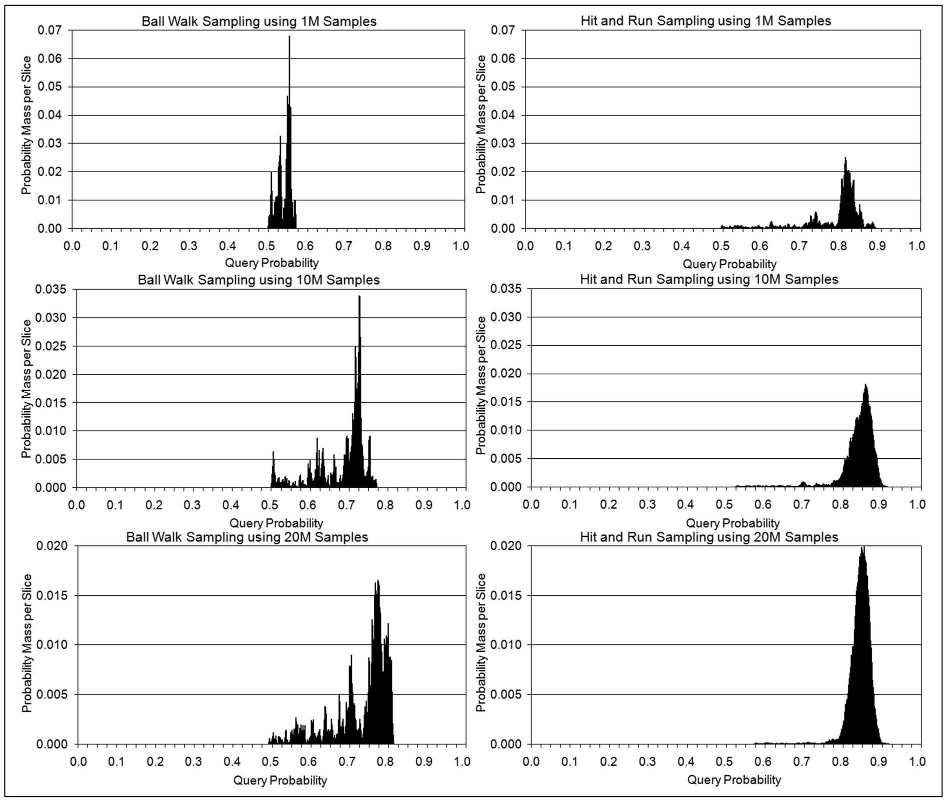

Figure 5: Histograms output by different runs of the Ball Walk (left) and Hit and Run (right) algorithms on the same PLP with 10 rules (341 variables in the LP) for different sample sizes. Note that the y axis has different scales at different sample sizes.

tervals associated with PLP queries can be inordinately wide, often giving very little information to the user about the truth or falsity of the query and, as illustrated in our stock example, making it difficult for the user to make decisions. Past approaches to this problem have been relatively *ad hoc*, arbitrarily choosing solutions in $LC(\Pi)$ that somehow correspond to some intuition of the researcher, such as maximal entropy. Such approaches are valid when the assumptions are valid in the application domain, but little or no effort has gone into verifying whether those assumptions are valid. Presumably the user will decide, but consider the feasibility of asking a stock analyst who has no idea what entropy is to decide whether maximal entropy is the right semantics for him.

In this paper, we solve this problem *without making any assumptions*, and at the same time providing a simple, graphical output to the user in the form of an easy to understand histogram. We do this by defining, for the first time, the unique notion of a *histogram answer* to a query Q w.r.t. a PLP Π. We show that the histogram answer computation problem is $\#P$-hard, and further show a close relationship between the problem of histogram answer computation and volume computation in convex polytopes. We provide an exact algorithm to compute histogram answers (which is expectedly inefficient because of the $\#P$-hardness result). We further develop an approximation algorithm Approx-HOPE that can work with any sampling method and evaluate it using two types of random walk sampling strategies: *Random Ball Walk* and *Hit and Run*. We develop an initial (small) prototype and quickly discover that Approx-HOPE combined with Hit and Run is much more efficient than with Random Ball Walk.

References

[1] APPLEGATE, D., AND KANNAN, R. Sampling and integration of near log-concave functions. In *ACM STOC* (New Orleans, USA, 1991), ACM, pp. 156–163.

[2] BIIELER, B., ENGE, A., AND FUKUDA, K. Exact volume computation for polytopes: A practical study. In *Polytopes: Combinatorics and Computation* (2000), Birkhauser.

[3] COHEN, J., AND HICKEY, T. Two algorithms for determining volumes of convex polyhedra. *Journal of the ACM 26*, 3 (1979), 401–414.

[4] DEKHTYAR, A., AND DEKHTYAR, M. I. Possible worlds semantics for probabilistic logic programs. In *ICLP* (2004), pp. 137–148.

[5] DYER, M., FRIEZE, A., AND KANNAN, R. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM 38*, 1 (1991), 1–17.

[6] DYER, M. E., AND FRIEZE, A. M. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing 17*, 5 (1988), 967–974.

[7] FAGIN, R., HALPERN, J. Y., AND MEGIDDO, N. A logic for reasoning about probabilities. *Information and Computation 87*, 1/2 (1990), 78–128.

[8] HAILPERIN, T. Probability logic. *Notre Dame J. of Formal Logic 25 (3)* (1984), 198–212.

[9] KHACHIYAN, L. The problem of calculating the volume of a polyhedron is enumerably hard. *Russian Mathematical Surveys 44*, 3 (1989), 199–200.

[10] KHULLER, S., MARTINEZ, M. V., NAU, D., SIMARI, G., SLIVA, A., AND SUBRAHMANIAN, V. Computing most probable worlds of action probabilistic logic programs: Scalable estimation for $10^{30,000}$ worlds. *AMAI 51*, 2–4 (2007), 295–331.

[11] LOVÁSZ, L., AND VEMPALA, S. Hit-and-run from a corner. In *ACM STOC* (Chicago, IL, USA, 2004), ACM, pp. 310–314.

[12] LOVÁSZ, L., AND VEMPALA, S. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences 72*, 2 (2006), 392–417.

[13] LUKASIEWICZ, T. Probabilistic logic programming. In *ECAI* (1998), pp. 388–392.

[14] LUKASIEWICZ, T., AND KERN-ISBERNER, G. Probabilistic logic programming under maximum entropy. *LNAI (ECSQARU-1999) 1638* (1999).

[15] MONTENEGRO, R., AND TETALI, P. Mathematical aspects of mixing times in markov chains. *Foundations and Trends in Theoretical Computer Science. 1*, 3 (2006), 237–354.

[16] NG, R. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *Journal of Automated Reasoning 10* (1993), 565–580.

[17] NG, R. T., AND SUBRAHMANIAN, V. S. Probabilistic logic programming. *Information and Computation 101*, 2 (1992), 150–201.

[18] NILSSON, N. Probabilistic logic. *Artificial Intelligence 28* (1986), 71–87.

[19] SHAPIRO, E. Y. Logic programs with uncertainties: A tool for implementing rule-based systems. In *IJCAI* (1983), pp. 529–532.

[20] VAN EMDEN, M. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming 4* (1986), 37–53.