ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-561

Probabilistic Preference Logic Networks

Thomas Lukasiewicz [1] **Maria Vanina Martinez** [1] **Gerardo I. Simari** [1]

Abstract. Reasoning about an entity's preferences (be it a user of an application, an individual targeted for marketing, or a group of people whose choices are of interest) has a long history in different areas of study. In this paper, we adopt the point of view that grows out of the intersection of databases and knowledge representation, where preferences are usually represented as strict partial orders over the set of tuples in a database or the consequences of a knowledge base. We introduce probabilistic preference logic networks (PPLNs), which flexibly combine such preferences with probabilistic uncertainty. Their applications are clear in domains such as the Social Semantic Web, where users often express preferences in an incomplete manner and through different means, many times in contradiction with each other. We show that the basic problems associated with reasoning with PPLNs (computing the probability of a world or a given query) are #P-hard, and then explore ways to make these computations tractable by: (i) leveraging results from order theory to obtain a polynomial-time randomized approximation scheme (FPRAS) under fixed-parameter assumptions; and (ii) studying a fragment of the language of PPLNs for which exact computations can be performed in fixed-parameter polynomial time.

1 Introduction

Interest in the Social Semantic Web has been growing recently as users continue to spend increasing amounts of time on platforms that allow sharing of different kinds of content, both conventional as well as user-generated. This poses a new challenge to knowledge representation and reasoning formalisms that are tasked with both modeling this kind of data as well as querying it in useful ways. One of the central aspects of this effort lies in reasoning about *preferences*.

Consider the case in which we want to model a user's food preferences: given the user's statements on his/her favorite social networking site, we have learned that (i) the user usually prefers tortellini over ravioli; (ii) with greater likelihood, the user prefers ravioli over lasagna; and (iii) the user is less likely to prefer ravioli over minestrone. We are interested in leveraging this incomplete and uncertain information to reason about cases for which no (direct) information is available—for instance, how likely is it that the user prefers tortellini over lasagna? How about tortellini over minestrone? Perhaps even more interestingly, what are the user's preferences regarding minestrone and lasagna, for which not even transitive preferences have been expressed? This situation is formalized as a probabilistic preference logic network (PPLN) in Example 5.

We would like to be able to model this kind of situation, which is characterized by: (i) the fact that we only have information on certain pairs of elements; and (ii) the uncertainty underlying the information provided—users are much more likely to express preferences that are subject to exceptions than ones that hold all the time. The main contributions of this paper are summarized as follows:

- We show that the main computational problems associated with reasoning with PPLNs (computing the probability of a world or a query in the form of a preference statement) are #P-hard.
- We leverage results from order theory to develop the anytimeQE-approx algorithm, a (fixed-parameter) fully polynomial-time randomized approximation scheme (FPRAS), in the form of an anytime algorithm, for computing the probability of a query.
- We study a fragment of the language of PPLNs, called k-decomposable PPLNs, for which exact computations can be performed in fixed-parameter polynomial time.

The rest of this paper is organized as follows. Section 2 introduces preliminaries on Markov random fields (the basis of the probabilistic semantics of PPLNs). In Section 3, we introduce PPLNs. Section 4 develops the concept of equivalence classes and studies its properties—this machinery is necessary to show how algorithms for counting linear extensions of strict partial orders can be leveraged in answering queries to PPLNs. Section 5 presents k-decomposable PPLNs, a subset of the full language that affords query answering in polynomial time assuming that both k and the size of the PPLN are bounded by a constant. Finally, Section 6 discusses related work, and Section 7 provides some concluding remarks.

2 Preliminaries on Markov Random Fields

We first recall the concept of Markov random field, on which the probabilistic semantics of PPLNs rests.

A Markov random field (MRF) is a probabilistic model that represents a joint probability distribution over a (finite) set of random variables $X = \{X_1, \ldots, X_n\}$. Each X_i may take on *values* from a finite *domain* $Dom(X_i)$. A *value* for $X = \{X_1, \ldots, X_n\}$ is a mapping $x\colon X \to \bigcup_{i=1}^{n} Dom(X_i)$ such that $x(X_i) \in Dom(X_i)$; the *domain* of X, denoted $Dom(X)$, is the set of all values for X. An MRF is similar to a Bayesian network (BN) in that it includes a graph $G = (V, E)$ in which each node corresponds to a variable. Differently from a BN, the graph is undirected; also, in an MRF, two variables are connected by an edge in G iff they are conditionally dependent. Furthermore, the model contains a *potential function* ϕ_j for each (maximal) clique in the graph; potential functions are non-negative real-valued functions of the values of the variables in each clique (called the *state* of the clique). Here, we assume the *log-linear* representation of MRFs, which involves defining a set of *features* of such states; a feature is a real-valued function of the state of a clique (we only consider binary features here). Given a value $x \in Dom(X)$ and a feature f_j for clique j, the probability distribution represented by an MRF is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j \cdot f_j(x)\right)$, where j ranges over the set of cliques in the graph G, and $w_j = \log \phi_j(x_{\{j\}})$

[1] Department of Computer Science, University of Oxford, UK; email: {thomas.lukasiewicz, vanina.martinez, gerardo.simari}@cs.ox.ac.uk.

(here, $x_{\{j\}}$ is the state of the j-th clique). The term Z is a normalization constant to ensure that $P(X=x) \in [0,1]$, and is given by $Z = \sum_{x \in Dom(X)} \exp\big(\sum_j w_j \cdot f_j(x)\big)$. Probabilistic inference in MRFs is intractable [23]; however, approximate inference mechanisms, such as Markov Chain Monte Carlo, have been developed and successfully applied in practice.

3 PPLNs: Probabilistic Preference Logic Networks

In this section, we introduce our knowledge representation formalism for expressing preferences under probabilistic uncertainty over a set of objects; its probabilistic semantics is based on MRFs.

Syntax. We denote by Δ and $\mathcal{R}$ the finite sets of constants and predicate symbols, respectively, in the language over which preferences are defined—i.e., we are interested in establishing preferences over the elements of the Herbrand base $\mathcal{H}$ induced by these sets.

Definition 1 A *possible world* is any permutation of the elements in the Herbrand base $\mathcal{H}$. We denote by $\mathcal{W}$ the set of all possible worlds.

Given atom a appearing in world λ, we use $pos(a,\lambda)$ to denote the natural number corresponding to the position of a in the sequence λ. Clearly, we have $|\mathcal{W}| = |\mathcal{H}|!$.

Intuitively, in the context of reasoning about preferences, the set of all possible worlds consists of all ways in which the elements in question can be (linearly) ordered—extending the formalism to contemplate more complex possible worlds (such as trees) is out of the scope of this paper, and will be tackled in future work.

The basic building block in the construction of our framework is the concept of *preference statement*, which is defined next.

Definition 2 A *(ground) preference statement* is either of the form: (i) $a \succ b$ (called *atomic preference statement* (or *(preference) atom*)), where a and b are atoms over Δ and $\mathcal{R}$, or (ii) $f \wedge g$, $f \vee g$, or $\neg f$, where f and g are (ground) preference statements. We denote by $\mathcal{A}_{Pref}$ the set of all preference atoms over Δ and $\mathcal{R}$. For $(a \succ b) \in \mathcal{A}_{Pref}$, statements $(b \succ a)$ and $\neg(a \succ b)$ are equivalent.

Note. For ease of presentation, we assume that all PPLNs are ground; extending the definition to non-ground preference statements is simple. This is without loss of generality, since groundings can be computed in polynomial time assuming bounded predicate arities.

Satisfaction of a preference statement ϕ by a world λ is defined as usual: (i) if $\phi = a \succ b$ such that $\phi \in \mathcal{A}_{Pref}$, then $\lambda \models \phi$ iff $a, b \in \lambda$ and $pos(a,\lambda) < pos(b,\lambda)$; (ii) if $\phi = \phi_1 \wedge \phi_2$, then $\lambda \models \phi$ iff $\lambda \models \phi_1$ and $\lambda \models \phi_2$; (iii) if $\phi = \phi_1 \vee \phi_2$, then $\lambda \models \phi$ iff $\lambda \models \phi_1$ or $\lambda \models \phi_2$; and (iv) if $\phi = \neg\phi'$, then $\lambda \models \phi$ iff $\lambda \not\models \phi'$.

Definition 3 A *probabilistic preference logic network* (PPLN) is a finite set P of pairs (F, w), where F is a preference statement over Δ and $\mathcal{R}$, and $w \in \mathbb{R} \cup \{\infty\}$. We refer to $|P|$ as the *size* of the PPLN.

The symbol "∞" denotes a number large enough to outweigh the values arising from other weights in the computation of probabilities and partition function in the MRF.

Semantics. In the following, we assume that there is an arbitrary (but fixed) order over the atoms in $\mathcal{H}$, denoted by the symbol "$<$". The probabilistic semantics of a PPLN P is given by the Markov random field that is defined as follows:

(i) For each $a, b \in \mathcal{H}$ such that $a < b$, there exists a (binary) node (a,b); the node's value is 1, if $a \succ b$ is true, and 0, otherwise;

(ii) one feature is defined for each statement in P, with value 1 iff the corresponding statement is true (and 0, otherwise); the weight of this feature is the weight associated with the statement in P; and

(iii) additional *infinite weight* features defined according to the following templates. For each $X, Y, Z \in \mathcal{H}$ such that $X < Y < Z$:

$$((X,Y)=1) \wedge ((Y,Z)=1) \Rightarrow (X,Z)=1;$$
$$((X,Y)=1) \wedge ((X,Z)=0) \Rightarrow (Y,Z)=0;$$
$$((X,Z)=0) \wedge ((Y,Z)=1) \Rightarrow (X,Y)=0;$$
$$((X,Y)=0) \wedge ((Y,Z)=0) \Rightarrow (X,Z)=0.$$

Recall that MRFs do not contain logical variables—these features are simply described in this way to show how they are built. Intuitively, these features impose a probability of zero on assignments of values to random variables that cause transitivity to be violated.

This characterization implies that the MRF contains $\binom{|\mathcal{H}|}{2}$ nodes, and an edge between nodes iff the preference atoms associated with such nodes appear together in a feature. Condition (iii) effectively provides a 1-to-1 mapping between PPLN worlds and value assignments to random variables in the underlying MRF with non-zero probability.

Example 4 Let $\mathcal{H} = \{a,b,c\}$ (where "$<$" is the lexicographical order), and $P = \{(b \succ a, 1), (b \succ c, 1)\}$ be a PPLN. The following table illustrates the mapping between the values of the variables in the induced MRF and the PPLN worlds (linear extensions).

(a,b)	(b,c)	(a,c)	Pairwise preferences	Linear extension
1	1	1	$a \succ b,\ b \succ c,\ a \succ c$	$\langle a,b,c\rangle$
1	1	0	$a \succ b,\ b \succ c,\ c \succ a$	–
1	0	1	$a \succ b,\ c \succ b,\ a \succ c$	$\langle a,c,b\rangle$
1	0	0	$a \succ b,\ c \succ b,\ c \succ a$	$\langle c,a,b\rangle$
0	1	1	$b \succ a,\ b \succ c,\ a \succ c$	$\langle b,a,c\rangle$
0	1	0	$b \succ a,\ b \succ c,\ c \succ a$	$\langle b,c,a\rangle$
0	0	1	$b \succ a,\ c \succ b,\ a \succ c$	–
0	0	0	$b \succ a,\ c \succ b,\ c \succ a$	$\langle c,b,a\rangle$

Out of $2^3 = 8$ possible truth assignments to MRF variables, we have $|\mathcal{H}|! = 6$ possible PPLN worlds— the other two (lines 2 and 7 above) violate the infinite weight features and thus have probability zero. ■

Thus, given this setup, we can compute probabilities over possible worlds analogously to Markov logic [22]—the probability of a world λ is given by the expression:

$$\Pr(\lambda) = \tfrac{1}{Z} \cdot \exp\big(\textstyle\sum_i w_i \cdot sat_i(\lambda)\big), \tag{1}$$

where $sat_i(x) = 1$ iff the formula F_i such that $(F_i, w_i) \in P$ is satisfied by λ (zero otherwise). The term Z is defined analogously as above for MRFs:

$$Z = \textstyle\sum_{\lambda \in \mathcal{W}} \exp\big(\sum_i w_i \cdot sat_i(\lambda)\big). \tag{2}$$

Queries. Computing the probability that a preference statement Q holds is then based on the worlds that Q satisfies: $\Pr(Q) = \sum_{\lambda \in \mathcal{W}, \lambda \models Q} \Pr(\lambda)$. The following is a simple example.

Example 5 Suppose a user has expressed the following preferences via the PPLN:

ϕ_1: (*tortellini* $\succ$ *ravioli*, 3), ϕ_2: (*ravioli* $\succ$ *lasagna*, 4.2),
ϕ_3: (*ravioli* $\succ$ *minestrone*, 2.1).

These formulas express that the user prefers tortellini over ravioli, ravioli over lasagna, and ravioli over minestrone, each with a corresponding weight. There are thus $4! = 24$ possible worlds (cf. Figure 1—initials used to save space).

	Worlds	Satisfies	Potential	Probability
λ_1	$\langle t,r,l,m\rangle$	ϕ_1,ϕ_2,ϕ_3	$3+4.2+2.1=9.3$	$e^{9.3}/Z$
λ_2	$\langle t,r,m,l\rangle$	ϕ_1,ϕ_2,ϕ_3	$3+4.2+2.1=9.3$	$e^{9.3}/Z$
λ_3	$\langle t,l,r,m\rangle$	ϕ_1,ϕ_3	$3+2.1=5.1$	$e^{5.1}/Z$
λ_4	$\langle t,l,m,r\rangle$	ϕ_1	3	e^{3}/Z
λ_5	$\langle t,m,r,l\rangle$	ϕ_1,ϕ_2	$3+4.2=7.2$	$e^{7.2}/Z$
λ_6	$\langle t,m,l,r\rangle$	ϕ_1	3	e^{3}/Z
λ_7	$\langle r,t,l,m\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$
λ_8	$\langle r,t,m,l\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$
λ_9	$\langle r,l,t,m\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$
λ_{10}	$\langle r,l,m,t\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$
λ_{11}	$\langle r,m,l,t\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$
λ_{12}	$\langle r,m,t,l\rangle$	ϕ_2,ϕ_3	$4.2+2.1=6.3$	$e^{6.3}/Z$

	Worlds	Satisfies	Potential	Probability
λ_{13}	$\langle l,t,r,m\rangle$	ϕ_1,ϕ_3	$3+2.1=5.1$	$e^{7.2}/Z$
λ_{14}	$\langle l,t,m,r\rangle$	ϕ_1	3	e^{3}/Z
λ_{15}	$\langle l,r,t,m\rangle$	ϕ_3	2.1	$e^{2.1}/Z$
λ_{16}	$\langle l,r,m,t\rangle$	ϕ_3	2.1	$e^{2.1}/Z$
λ_{17}	$\langle l,m,t,r\rangle$	ϕ_1	3	e^{3}/Z
λ_{18}	$\langle l,m,r,t\rangle$	–	0	e^{0}/Z
λ_{19}	$\langle m,t,r,l\rangle$	ϕ_1,ϕ_2	$3+4.2=7.2$	$e^{7.2}/Z$
λ_{20}	$\langle m,t,l,r\rangle$	ϕ_1	3	e^{3}/Z
λ_{21}	$\langle m,r,t,l\rangle$	ϕ_2	4.2	$e^{7.2}/Z$
λ_{22}	$\langle m,r,l,t\rangle$	ϕ_2	4.2	$e^{7.2}/Z$
λ_{23}	$\langle m,l,t,r\rangle$	ϕ_1	3	e^{3}/Z
λ_{24}	$\langle m,l,r,t\rangle$	–	0	e^{0}/Z

Figure 1. The value of Z (computed as the sum of all the numbers in the "Potential" columns) is approximately 28,422.59.

The probability of a query is given by summing the probabilities of the worlds that satisfy it—for instance, the probability that minestrone will be preferred over lasagna is computed as:

$$\begin{aligned}\Pr(m \succ l) \;&=\; \Pr(\lambda_2)+\Pr(\lambda_5)+\Pr(\lambda_6)+\Pr(\lambda_8)+\\ &\quad\ \Pr(\lambda_{11})+\Pr(\lambda_{12})+\Pr(\lambda_{19})+\Pr(\lambda_{20})+\\ &\quad\ \Pr(\lambda_{21})+\Pr(\lambda_{22})+\Pr(\lambda_{23})+\Pr(\lambda_{24})\\ &\approx\; 0.5434. \qquad \blacksquare\end{aligned}$$

Complexity Results. The following theorem shows that computing the probability of a preference statement is #P-hard.

Theorem 1 *Given a PPLN P and a preference statement S relative to P, computing the probability of S is #P-hard.*

Proof. Let Φ be a 3SAT formula in CNF with variables $V_1,\ldots,V_n$. Create a PPLN with $\mathcal{H}=\{v_1,\ldots,v_n\}\cup\{\bar v_1,\ldots,\bar v_n\}$ and a single preference statement S built in the following manner: for each clause $Q_1\vee Q_2\vee Q_3$ in Φ create a corresponding clause $P_1\vee P_2\vee P_3$, where $P_i=(v_j\succ\bar v_j)$, if $Q_i=V_j$, and $P_i=(\bar v_j\succ v_j)$, if $Q_i=\neg V_j$, for every $i\in\{1,2,3\}$ and some $j\in\{1,\ldots,n\}$. The PPLN then consists of a single preference statement comprised of the conjunction S of all such clauses, with weight 1.

Given a world λ, we can define a corresponding world λ_{SAT} defined as a truth assignment to the variables $V_1,\ldots,V_n$ as follows: V_j is true iff $pos(v_j,\lambda)<pos(\bar v_j,\lambda)$. This implies that $\lambda\models S\Leftrightarrow\lambda_{SAT}\models\Phi$. Therefore, if we compute the probability of S relative to PPLN P, we get: $\Pr(S)=\sum_{\lambda\models S}e^1/Z$, with $Z=\sum_{\lambda\in\mathcal{W}}\exp(1\cdot n(\lambda))$, where $n(\lambda)=1$ if $\lambda\models S$ and $n(\lambda)=0$ otherwise. But this means that: $\Pr(S)=\frac{e\cdot k}{e\cdot k+((2n)!-k)}=p$, where k is the number of worlds in $\mathcal{W}$ that satisfy S. Thus, solving for k, we conclude that $\Pr(S)=p$ iff there are $k=\frac{(2n)!}{(p-1)\cdot e+1}$ worlds in $\mathcal{W}$ that satisfy S, which means that Φ has precisely $k\cdot 2^n/(2n)!$ satisfying assignments. This concludes the reduction from #3SAT. □

The following result can be shown using a reduction similar to the one used in the proof of Theorem 1.

Theorem 2 *Given a PPLN P and a world λ relative to P, computing the probability of λ is #P-hard.*

As in similar probabilistic knowledge representation formalisms [19, 22], one way to tackle this intractability is by means of random sampling methods, as described next.

Markov Chain Monte Carlo Approaches. Since the semantics of PPLNs is given by MRFs, the most basic approach to perform approximate computations over our model is to directly construct the corresponding MRF and apply the standard MCMC methods that have been developed and studied for decades now (see [9] for a survey of such methods). Such an approach, however, is likely to be suboptimal, since the MRF contains $\binom{n}{2}$ nodes whenever the PPLN contains n atoms and, apart from this, the state space is broken into many pieces by the constraints described in point (iii) above, so the connectedness condition required for MCMC is violated—methods such as slice sampling [17] are thus needed.

A more direct application of MCMC methods would involve sampling directly from the set of PPLN worlds (permutations of atoms). Setting up a Markov Chain in such a way that its stationary distribution allows us to sample worlds from a distribution that follows the one described by Equations 1 and 2 is the topic of future work—related work in order theory has approached the task in the non-probabilistic setting with promising results [20, 6].

4 Approximations via Approximate Counting

We now investigate ways in which the intractability of computing entailment probabilities can be tackled by leveraging algorithms for counting linear extensions (topological sortings) of strict partial orders (referred to as SPOs from now on), which are simply irreflexive and transitive binary relations. To do this, we first present some basic machinery that is based on the division of such linear extensions into *equivalence classes*; this suggestion is not new in the literature on probabilistic reasoning—it has been suggested in approaches ranging from basic probabilistic logic [18] to probabilistic ontology languages [13]. In the following, given PPLN P and world λ, $sat(\lambda,P)$ denotes the set of statements in P that are satisfied by λ.

Definition 6 Let P be a PPLN. Worlds λ_1 and λ_2 are *equivalent* relative to P, denoted $\lambda_1\sim_P\lambda_2$, iff $sat(\lambda_1,P)=sat(\lambda_2,P)$.

We use $equiv(P)=\{C_1,C_2,\ldots,C_N\}$ to denote the set of equivalence classes induced by equivalence relation $\sim_P$. Essentially, equivalence classes can be characterized by a preference statement composed of the conjunction of each preference statement in P either as it appears in P or negated—note that this directly implies that, in the worst case, there are an exponential number of such classes.

An interesting and useful property of equivalence classes is stated next. It arises from the fact that the probability of a world is determined by what formulas it satisfies; since equivalent worlds satisfy the same formulas, the result follows.

Theorem 3 *If $\lambda_1\sim_P\lambda_2$, then $\Pr(\lambda_1)=\Pr(\lambda_2)$.*

The following example illustrates the equivalence class approach.

Class	PPLN Worlds	Aggregate Probability
C_1	$\{\lambda_1, \lambda_2\}$	$2 * e^{9.3}/Z$
C_2	$\{\lambda_3, \lambda_{13}\}$	$2 * e^{5.1}/Z$
C_3	$\{\lambda_4, \lambda_6, \lambda_{14}, \lambda_{17}, \lambda_{20}, \lambda_{23}\}$	$6 * e^{3}/Z$
C_4	$\{\lambda_5, \lambda_{19}\}$	$2 * e^{7.2}/Z$
C_5	$\{\lambda_7, \lambda_8, \lambda_9, \lambda_{10}, \lambda_{11}, \lambda_{12}\}$	$6 * e^{6.3}/Z$
C_6	$\{\lambda_{15}, \lambda_{16}\}$	$2 * e^{2.1}/Z$
C_7	$\{\lambda_{18}, \lambda_{24}\}$	$2 * e^{0}/Z$
C_8	$\{\lambda_{21}, \lambda_{22}\}$	$2 * e^{4.2}/Z$

Figure 2. Equivalence classes of Example 7.

Example 7 Consider again the PPLN from Example 5. The equivalence classes of possible worlds are determined by the following preference statements (names abbreviated to single letters):

$$\begin{array}{llllll}
C_1: & (t \succ r) & \wedge & (r \succ l) & \wedge & (r \succ m), \\
C_2: & (t \succ r) & \wedge & \neg(r \succ l) & \wedge & (r \succ m), \\
C_3: & (t \succ r) & \wedge & \neg(r \succ l) & \wedge & \neg(r \succ m), \\
C_4: & (t \succ r) & \wedge & (r \succ l) & \wedge & \neg(r \succ m), \\
C_5: & \neg(t \succ r) & \wedge & (r \succ l) & \wedge & (r \succ m), \\
C_6: & \neg(t \succ r) & \wedge & \neg(r \succ l) & \wedge & (r \succ m), \\
C_7: & \neg(t \succ r) & \wedge & \neg(r \succ l) & \wedge & \neg(r \succ m), \\
C_8: & \neg(t \succ r) & \wedge & (r \succ l) & \wedge & \neg(r \succ m).
\end{array}$$

The composition of each class is shown in Figure 2. Here, all classes are non-empty; this is not always so, as the statements describing a class may be unsatisfiable, e.g., $(a \succ b) \wedge (b \succ c) \wedge \neg(a \succ c)$. ■

Another useful property of equivalence classes is that we can decide their emptiness in polynomial time.

Theorem 4 *Given a PPLN P and an equivalence class C relative to P, deciding whether $|C| = 0$ can be done in polynomial time.*

Intuitively, a class induces a directed graph—constructing it and testing for cycles suffices to decide emptiness of the class, and this can be done in polynomial time.

Theorems 3 and 4 suggest that, if we can count the number of worlds that satisfy the preference statement describing a class (its linear extensions), we are well on our way to saving steps in the computation of probabilities of queries—instead of iterating through all $n!$ worlds, we can inspect the equivalence classes and simply compute: $Pr(Q) = \sum_{C \in \mathit{equiv}(P'),\, C \models Q} \big(|C| \cdot \Pr(C)\big)$, where $P' = P \cup \{(Q, 0)\}$ (the weight here is irrelevant). Unfortunately, the problem of counting linear extensions of an SPO is #P-hard as well [5]. There are, however, several algorithms developed in the order theory literature that approximate this computation [5, 20, 6, 1].

The best known algorithm to date is called TPA [1]—it is a fully polynomial randomized approximation scheme (FPRAS) that, given an SPO O, approximates $\mathcal{L}(O)$ (the number of linear extensions of O) to within a factor of $1 + \epsilon$ with probability at least $1 - \delta$, using an expected number of random bits and comparisons, in time at most $O((\ln \mathcal{L}(O))^2 \cdot n^3 \cdot (\ln n) \cdot \epsilon^{-2} \cdot \ln(1/\delta))$—we refer to this expression as *Cost*(*TPA*). In the worst case, $\ln(\mathcal{L}(O))$ is $O(n \cdot \ln n)$, and the worst case complexity is the same as in the first FPRAS presented in [5] (the authors of the TPA approach claim, however, that the hidden constants in the worst case are about 1,000 times smaller). On the other hand, in cases in which $\mathcal{L}(O)$ is small compared to $n!$, the TPA algorithm is much faster.

Thus, we can compute the probability of a query by adding it to the set of preference statements and defining the equivalence classes over the set of resulting statements—if *Cost*(*TPA*) $\cdot\, O(2^{|P|+1})$ is "smaller than" $O(n!)$, this approach will be faster than ones based on iterating through all possible worlds. There is one more result that can be leveraged in designing an algorithm along these lines.

Class	PPLN Worlds	Score	$C \models (m \succ l)$?
C_1	$\{\lambda_1\}$	$1 * e^{9.3}$	No
C'_1	$\{\lambda_2\}$	$1 * e^{9.3}$	Yes
C_4	$\emptyset$	$0 * e^{7.2}$	No
C'_4	$\{\lambda_5, \lambda_{19}\}$	$2 * e^{7.2}$	Yes
C_5	$\{\lambda_7, \lambda_9, \lambda_{10}\}$	$3 * e^{6.3}$	No
C'_5	$\{\lambda_8, \lambda_{11}, \lambda_{12}\}$	$3 * e^{6.3}$	Yes

Figure 3. Classes from Example 8, sorted by score.

Theorem 5 *Given a PPLN P, the $\sim_P$-equivalence classes can be listed in output-polynomial time in descending order of probability of their individual worlds.*

This result follows from the fact that the values arising from the second term in Eq. 1, i.e., without dividing by Z (let's call this the *score* of a world or class) can be enumerated in descending order by sorting the formulas in P in descending order of weight and iterating through the truth values of each formula to generate all possible classes.

This insight is the basis of Algorithm anytimeQE-approx (Figure 4), which approximates the probability with which a PPLN entails a query. The main while-loop iterates through the classes in descending order of probability as discussed above; for each non-empty one, it runs the approximation algorithm to compute its size and then checks if the worlds in the class entail the query or not, updating the corresponding score mass. Whenever the loop is interrupted (or finished), the algorithm outputs the current estimation of the probability value. The following shows this process over the running example.

Example 8 Returning to Example 7 and Figure 2, note that the statements that are not negated in the class are the ones that contribute to the probability. Thus, to iterate in descending order of probability we start from the class with all non-negated statements and then iteratively negate the statements with lowest weight in the PPLN, the two statements with lowest weight, and so forth. Figure 3 shows the top three most probable classes produced in this manner, where the classes are numbered as in Figure 2, but each is split into two (C_i and C'_i) according to their satisfaction of the query.

Suppose that we run Algorithm 4 on this PPLN, with the query $(m \succ l)$ ("is minestrone preferred over lasagna?"); suppose further that, for the purpose of this example, the TPA approximations yield exact counts, and that the stopping predicate causes the algorithm to inspect six classes. The algorithm updates the values of *score-pos* and *score-neg*, which at the end of the run receive values of approximately $15,250.59$ and $12,571.73$—this causes the algorithm to return $15,250.59/(15250.59 + 12571.73) \approx 0.5481$; recall from Example 5 that the exact value is ≈ 0.5434. ■

The next result states two interesting properties of our algorithm.

Theorem 6 *Algorithm* anytimeQE-approx(P, Q, *stopPred*, ϵ, δ) *enjoys the following properties:*

(a) *Its running time is $O(m * poly(n))$, where n is the size of the input, and m is the number of equivalence classes inspected before the stopping condition becomes true.*

(b) *If the algorithm inspects all classes, and $|P|$ is considered fixed, the algorithm is a fully polynomial-time approximation scheme—we are guaranteed an approximation of* $\Pr(Q)$ *such that:*

$$(1+\epsilon)^{-1} \cdot 2^{|P|+1} \leqslant \mathit{approx}(Q) \leqslant (1+\epsilon) \cdot 2^{|P|+1}$$

with probability at least $(1-\delta)^{2^{|P|+1}}$.

Algorithm anytimeQE-approx$(P, Q, \mathit{stopPred}, \epsilon, \delta)$
// P is a ground PPLN, Q is a preference statement query

score-pos:= 0; // Score mass of worlds that satisfy Q
score-neg:= 0; // Score mass of worlds that do not satisfy Q
$i := 1$;
while $\left(i \leqslant 2^{|P|+1}\right)$ and !*stopPred* do begin
// i ranges over classes of possible worlds
    C:= *compNextClass*(P, Q, i);
    i++;
    if C is not empty then
        size-C:= *TPA-approx*(C, ϵ, δ);
        if $C \models Q$ then
            score-pos:= *score-pos* + $\left(\textit{size-C} \cdot \Pr(C)\right)$;
        else
            score-neg:= *score-neg* + $\left(\textit{size-C} \cdot \Pr(C)\right)$;
    end;
end;
return *score-pos*$/\left(\textit{score-pos} + \textit{score-neg}\right)$.

Figure 4. Probabilistic preference query answering leveraging the TPA linear extension counting algorithm.

5 Tractable Cases via Exact Counting

In this section, we explore conditions under which the equivalence classes from Section 4 can be used in cases for which exact linear extension counting algorithms are guaranteed to run in polynomial time. Though several restrictions on SPOs have been studied under which counting can be done tractably [16], the use of equivalence classes means that the restrictions cannot be guaranteed to hold in general—the following example illustrates why this happens.

Example 9 Let $P = \{(a \succ b, 1), (b \succ c, 1), (c \succ d, 1)\}$, where f_1, f_2, and f_3 are the respective atomic preference statements and $\mathcal{H} = \{a, b, c, d, e, f\}$. It is very simple to count the linear extensions of the SPOs that arise from many of the equivalence classes, such as $f_1 \wedge f_2 \wedge f_3$ (linear order) and $\neg f_1 \wedge f_2 \wedge f_3$ (tree). However, the class given by $f_1 \wedge \neg f_2 \wedge f_3$ yields an SPO that is precisely the forbidden substructure of the so-called N-free orders [16]—such violation destroys the tractability guarantees. ■

Example 9 shows us how restrictions that hold for one equivalence class easily break down for others—the main cause of this is that negation of preference statements can have all kinds of effects on the induced SPO. One possibility that remains open, however, is that of enforcing a constraint over all equivalence classes that allows us to leverage a property of the class of *decomposable partial orders* (DPOs), which were introduced in [8].

Definition 10 A PPLN P is *k-decomposable* iff for each C in *equiv*(P), all connected components of the directed acyclic graph induced by C have size at most k.

DPOs generalize *series-parallel* partial orders [16] (equivalent to the N-free orders mentioned in Example 9), which can be characterized by means of two composition operations (series and parallel). In particular, we are interested in parallel composition, denoted $P \,||\, Q$ for SPOs P and Q—the key to leveraging the condition in Definition 10 lies in that parallel (de)composition leads to a simple recursive formula for counting linear extensions: $\mathcal{L}(O_1 \,||\, O_2) =$

$$\mathcal{L}(O_1) \cdot \mathcal{L}(O_2) \cdot \binom{|O_1| + |O_2|}{|O_1|} = \mathcal{L}(O_1) \cdot \mathcal{L}(O_2) \cdot \frac{\left(|O_1| + |O_2|\right)!}{|O_1|! \cdot |O_2|!}. \quad (3)$$

The k-decomposition property implies that all induced SPOs have the so-called bounded decomposition width property, which is known to admit counting algorithms in $O(n^k)$ [25, 26, 16]—we refer to this as the "Steiner bound" and, for the purposes of this paper, assume we have such an algorithm. We can thus count the number of linear extensions for a given class in $O(n^k)$; this leads to the result:

Theorem 7 *Let P be a k-decomposable PPLN, and Q be a preference statement. Then,* $\Pr(Q)$ *is computable in time* $O\left(2^{|P|} \cdot \frac{n}{k} \cdot n^k\right)$.

This is a direct result of running the Steiner bound algorithm for each of the $\frac{n}{k}$ fragments of the decomposed PPLN; the term $2^{|P|}$ arises from the computation of the equivalence classes. The main objective of k-decomposable PPLNs is, however, to assume that k is bounded.

Algorithm anytimeDec**.** These ideas can be implemented in an algorithm that follows the same basic outline as anytimeQE-approx—we discuss its details without presenting the pseudocode for reasons of space. As before, the equivalence classes are inspected in descending order of probability; for each one, the k components of the arising SPO are inspected in turn. An additional flag "*testTrac*" can be added as a parameter to the algorithm indicating whether or not additional effort should be spent on testing whether such SPO belongs to a class that allows tractable counting of linear extensions; for instance, series-parallel/N-free orders can be recognized in $O(n^2)$ time, and their number of linear extensions can be computed in the same time. If the tractability test fails, the Steiner bound ($O(n^k)$) algorithm can be used for this class instead. (Note that such a flag could also be added to anytimeQE-approx.) After the sizes of the k components are computed, the algorithm iteratively applies Equation 3 to obtain the final count for the class. Updates to accumulators for positive and negative scores are done as before.

The following is a simple extension of the running example to show how this algorithm works.

Example 11 Let P' be the PPLN from the running example, with the addition of the statement $(g \succ c, 2)$, meaning that gnocchi are preferred over canneloni. Note that the DAG induced by all preference statements in P' has two connected components, where the largest has four nodes—P' is thus 4-decomposable.

Consider now the equivalence class determined by $(t \succ r) \wedge (r \succ l) \wedge (r \succ m) \wedge \neg(g \succ c)$. We know from Figure 2 that two worlds satisfy $(t \succ r) \wedge (r \succ l) \wedge (r \succ m)$, while it is easy to see that only one satisfies $\neg(g \succ c)$. Then, the total number of worlds satisfying our class is given by: $2 \cdot 1 \cdot \frac{(4+2)!}{4! \cdot 2!} = 30$. ■

Finally, we have the following result regarding our algorithm.

Theorem 8 *Given k-decomposable PPLN P, Algorithm* anytimeDec*(P, Q, stopPred, testTrac) enjoys the following properties:*

(a) *Its worst case running time is $O(m \cdot \frac{n}{k} \cdot \mathit{poly}(n))$, where n is the size of the input, and m is the number of equivalence classes inspected before the stopping condition becomes true.*

(b) *If it inspects all classes, it returns an exact result for* $\Pr(Q)$.

Theorems 6 and 8 tell us that, if the size of the PPLN can be bounded by a constant, both algorithms are fixed-parameter polynomial-time.

6 Related Work

There has been a wide variety of work in the study and modeling of preferences in areas as varied as philosophy, logic, and economics.

In the philosophical tradition, preferences are usually expressed over mutually exclusive "outcomes", such as truth assignments to formulas. Using this interpretation of preferences, [2] aims at bridging the gap between several formalisms from the AI community such as CP-nets and those studied traditionally in philosophy. From the point of view of uncertainty, there is not a large body of work focusing on combining the two; the probabilistic extensions to CP-nets in [7, 3] are perhaps the closest in spirit to our work, but they follow the CP-net model of preferences (and probabilities) over possible worlds in the form of truth assignments to logical formulas.

Our work is most closely aligned with the database community's perspective on the problem, where preferences are established between tuples (which correspond to ground atoms here) and in general specify how results to queries should be sorted. The database tradition has been studied for almost three decades, since the seminal work of [12]; see [24] for a survey of notable works in this line. More or less in parallel, work has also been carried out in the intersection between databases and knowledge representation and reasoning, such as preference logic programs [10], incorporation of preferences in formalisms such as answer set programs [4], and answering k-rank queries in ontological languages [14]. Apart from [15], where probabilistic ontologies are assumed to yield a preference model that may be in conflict with another (user-provided) model, we are aware of no other work on combining probabilistic reasoning with logical representations such as the one used here.

The preference networks in [27] are based on establishing an MRF defined over random variables corresponding to users' ratings of items—the main goal is to obtain probabilities for user-item pairings that have not been observed, in order to issue recommendations based on previously rated items. Similar in spirit is [11] on modeling users with statistical preference models; their goal is to derive the probability that an item will be preferred by a user based on information regarding previous item selections. These approaches are therefore ad hoc for this kind of problem, instead of allowing general knowledge about preferences to be specified, as in PPLNs. Learning PPLNs is an important next step, and perhaps these models can be leveraged in effectively accomplishing this highly difficult task.

Finally, the kind of knowledge expressed in PPLNs can certainly be expressed in other formalisms, such as the probabilistic logic programs of [21], among others. But PPLNs are better suited to representing incomplete and uncertain preferences, since their semantics is based on linear orders as possible worlds, allowing specific methods to be applied. Future work involves studying techniques leveraged in related formalisms to enhance the scalability of PPLNs.

7 Summary and Outlook

In this paper, we have proposed PPLNs, a novel integration of a preference representation formalism in the tradition of databases (where preferences are expressed between tuples or ground atoms) with probabilistic uncertainty. Similar to Markov logic, the semantics is based on an underlying Markov random field; however, in PPLNs, possible worlds consist of permutations of the elements of interest rather than truth assignments. We argue that this basic difference merits rethinking how answers to probabilistic queries are computed, and we therefore studied how results from order theory can inform our effort. As a result, we provided an approximation algorithm and an exact algorithm that is fixed-parameter tractable; in addition, both algorithms were designed to work in an anytime fashion.

Future work in this line includes how PPLNs can be learned from real-world data, as well as testing our algorithms on such data to identify how greater tractability can be achieved in practice.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J0083-46/1 ("PrOQAW: Probabilistic Ontological Query Answering on the Web"), by the European Research Council (FP7/2007-2013)/ERC grant 246858 ("DIADEM"), by an EU (FP7/2007-2013) Marie-Curie Intra-European Fellowship, and by a Yahoo! Research Fellowship.

REFERENCES

[1] J. Banks, S. Garrabrant, M.L. Huber, and A. Perizzolo, 'Using TPA to count linear extensions', *ArXiv e-prints*, (2010).

[2] M. Bienvenu, J. Lang, and N. Wilson, 'From preference logics to preference languages, and back', in *Proc. of KR*, pp. 214–224, (2010).

[3] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini, 'Probabilistic conditional preference networks', in *Proc. of UAI*, pp. 72–81, (2013).

[4] G. Brewka, 'Preferences, contexts and answer sets', in *Proc. of ICLP*, p. 22, (2007).

[5] G. Brightwell and P. Winkler, 'Counting linear extensions is #P-complete', in *Proc. of STOC*, pp. 175–181, (1991).

[6] R. Bubley and M. Dyer, 'Faster random generation of linear extensions', *Discrete Mathematics*, **201**(1/3), 81–88, (1999).

[7] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K.B. Venable, 'Updates and uncertainty in CP-nets', in *Proc. of AUS-AI*, pp. 301–312, (2013).

[8] T. Gallai, 'Transitiv orientierbare Graphen', *Acta Mathematica Academiae Scientiarum Hungarica*, **18**(1/2), 25–66, (1967).

[9] W. R. Gilks, *Markov Chain Monte Carlo In Practice*, Chapman and Hall/CRC, 1999.

[10] K. Govindarajan, B. Jayaraman, and S. Mantha, 'Preference logic programming', in *Proc. of ICLP*, pp. 731–745, (1995).

[11] S. Jung, J. Hong, and T. Kim, 'A statistical model for user preference', *TKDE*, **17**(6), 834–843, (2005).

[12] M. Lacroix and P. Lavency, 'Preferences: Putting more knowledge into queries', in *Proc. of VLDB*, pp. 1–4, (1987).

[13] T. Lukasiewicz, M.V. Martinez, G. Orsi, and G.I. Simari, 'Heuristic ranking in tightly coupled probabilistic description logics', in *Proc. of UAI*, pp. 554–563, (2012).

[14] T. Lukasiewicz, M.V. Martinez, and G.I. Simari, 'Preference-based query answering in Datalog+/– ontologies', in *Proc. of IJCAI*, pp. 1017–1023, (2013).

[15] T. Lukasiewicz, M.V. Martinez, and G.I. Simari, 'Preference-based query answering in probabilistic Datalog+/– ontologies', in *Proc. of ODBASE*, pp. 501–518, (2013).

[16] R.H. Möhring, 'Computationally tractable classes of ordered sets', in *Algorithms and Order*, ed., Ivan Rival, NATO ASI series / C: NATO ASI series, pp. 105–193. Kluwer Academic Publishers, (1989).

[17] R.M. Neal, 'Slice sampling', *Annals of Statistics*, **31**, 705–767, (2003).

[18] N. Nilsson, 'Probabilistic logic', *Artif. Intell.*, **28**, 71–87, (1986).

[19] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[20] G. Pruesse and F. Ruskey, 'Generating linear extensions fast', *SIAM J. Comput.*, **23**(2), 373–386, (1994).

[21] L. De Raedt, A. Kimmig, and H. Toivonen, 'Problog: A probabilistic Prolog and its application in link discovery', in *Proc. of IJCAI*, pp. 2462–2467, (2007).

[22] M. Richardson and P. Domingos, 'Markov logic networks', *Mach. Learn.*, **62**(1/2), 107–136, (2006).

[23] Dan Roth, 'On the hardness of approximate reasoning', *Artif. Intell.*, **82**(1/2), 273–302, (1996).

[24] K. Stefanidis, G. Koutrika, and E. Pitoura, 'A survey on representation, composition and application of preferences in database systems', *ACM TODS*, **36**(3), 19:1–19:45, (2011).

[25] G. Steiner, 'On computing the information theoretic bound for sorting: Counting the linear extensions of posets', Technical report, Hamilton, ON, Canada, (1987).

[26] G. Steiner, 'On the information theoretic bound for sorting: Balancing the linear extensions of posets', Technical report, Hamilton, ON, Canada, (1987).

[27] T.T. Truyen, D.Q. Phung, and S. Venkatesh, 'Preference networks: Probabilistic models for recommendation systems', in *Proc. of AusDM*, pp. 195–202, (2007).