Symbolic Parameter Learning in Probabilistic Answer Set Programming

DAMIANO AZZOLINI

Department of Environmental and Prevention Sciences, University of Ferrara, Ferrara, Italy
(e-mail: damiano.azzolini@unife.it*)*

ELISABETTA GENTILI

Department of Engineering, University of Ferrara, Ferrara, Italy
(e-mail: elisabetta.gentili1@unife.it*)*

FABRIZIO RIGUZZI

Department of Mathematics and Computer Science, University of Ferrara, Ferrara, Italy
(e-mail: fabrizio.riguzzi@unife.it*)*

Abstract

Parameter learning is a crucial task in the field of Statistical Relational Artificial Intelligence: given a probabilistic logic program and a set of observations in the form of interpretations, the goal is to learn the probabilities of the facts in the program such that the probabilities of the interpretations are maximized. In this paper, we propose two algorithms to solve such a task within the formalism of Probabilistic Answer Set Programming, both based on the extraction of symbolic equations representing the probabilities of the interpretations. The first solves the task using an off-the-shelf constrained optimization solver while the second is based on an implementation of the Expectation Maximization algorithm. Empirical results show that our proposals often outperform existing approaches based on projected answer set enumeration in terms of quality of the solution and in terms of execution time.

Keywords: probabilistic answer set programming, statistical relational artificial intelligence, parameter learning, optimization

1 Introduction

Statistical Relational Artificial Intelligence (StarAI) (Raedt et al., 2016) is a subfield of Artificial Intelligence aiming at describing complex probabilistic domains with interpretable languages. Such languages are, for example, Markov Logic Networks (Richardson and Domingos, 2006), Probabilistic Logic Programs (De Raedt et al., 2007; Riguzzi, 2022), and Probabilistic Answer Set Programs (Cozman and Mauá, 2020). Here, we focus on the last. Within StarAI, there are many problems that can be considered such as probabilistic inference, MAP inference, abduction, parameter learning, and structure

learning. In particular, the task of parameter learning requires, given a probabilistic logic program and a set of observations (often called interpretations), tuning the probabilities of the probabilistic facts such that the likelihood of the observation is maximized. This is often solved by means of Expectation Maximization (EM) (Bellodi and Riguzzi, 2013; Dries et al., 2015; Azzolini et al., 2024) or Gradient Descent (Gutmann et al., 2008).

Recently, Azzolini and Riguzzi (2021) proposed to extract a symbolic equation for the probability of a query posed to a probabilistic logic program. From that equation, it is possible to model and solve complex constrained problems involving the probabilities of the facts (Azzolini, 2023).

In this paper, we propose two algorithms to learn the parameters of probabilistic answer set programs. Both algorithms are based on the extraction of symbolic equations from a compact representation of the interpretations, but they differ in how they solve the problem. The first casts parameter learning as a nonlinear constrained optimization problem and leverages off-the-shelf solvers, thus bridging the area of probabilistic answer set programming with constrained optimization, while the second solves the problem using EM. Empirical results, also against an existing tool to solve the same task, on four different datasets with multiple configurations show the proposal based on constrained optimization is often significantly faster and more accurate w.r.t. the other approaches.

The paper is structured as follows: Section 2 discusses background knowledge, Section 3 proposes novel algorithms to solve the parameter learning task, that are tested in Section 5. Section 4 surveys related works and Section 6 concludes the paper.

2 Background

An answer set program is composed by a set of normal rules of the form $h :- b_0, \ldots, b_m,\ not\ c_0, \ldots,\ not\ c_n$ where h, the b_is, and the c_is are *atoms*. h is called *head* while the conjunction of literals after the ":−" symbol is called *body*. A rule without a head is called *constraint* while a rule without a body is called *fact*. The semantics of an answer set program is based on the concept of *stable model* (Gelfond and Lifschitz, 1988), often called *answer set*. The set of all possible ground atoms for a program P is called *Herbrand base* and denoted with B_P. The grounding of a program P is obtained by replacing variables with constants in B_P in all possible ways. An *interpretation* is a subset of atoms of B_P and it is called *model* if it satisfies all the groundings of P. An *answer set* I of P is a minimal model under set inclusion of the reduct of P w.r.t. I, where the reduct w.r.t. I is obtained by removing from P the rules whose body is false in I.

2.1 Probabilistic answer set programming

We consider the Credal Semantics (CS) (Lukasiewicz, 2005; Cozman and Mauá, 2016; Mauá and Cozman, 2020) that associates a meaning to Answer Set Programs extended with probabilistic facts (De Raedt et al., 2007) of the form $p :: a$ where p is the probability associated with the atom a. Intuitively, such notation means that the fact a is present in the program with probability p and absent with probability $1 - p$. These programs are called Probabilistic Answer Set Programs (PASP, and we use the same acronym to also

indicate Probabilistic Answer Set Programming – the meaning will be clear from the context). A selection for the presence or absence of each probabilistic fact defines a world w whose probability is $P(w) = \prod_{a \in w} p \cdot \prod_{a \notin w}(1-p)$, where with $a \in w$ we indicate that a is present in w and with $a \notin w$ that a is absent in w. A program with n probabilistic facts has 2^n worlds. Let us indicate with W the set of all possible worlds. Each world is an answer set program and it may have zero or more answer sets but the CS requires at least one. If this holds, the probability of a query q (a conjunction of ground literals) is defined by a lower and an upper bound. A world w contributes to both the lower and upper probability if each of its answer sets contains the query, that is it is a cautious consequence. If only some answer sets contain the query, that is it is a brave consequence, w only contributes to the upper probability. If the query is not present, we have no contribution to the probability bounds from w. In formulas,

$$P(q) = [\underline{P}(q), \overline{P}(q)] = [\sum_{w_i \in W | \forall m \in AS(w_i), \ m \models q} P(w_i), \sum_{w_i \in W | \exists m \in AS(w_i), \ m \models q} P(w_i)]. \quad (1)$$

The conditional probability of a query q given evidence e, also in the form of conjunction of ground literals, is (Cozman and Mauá, 2020):

$$\underline{P}(q \mid e) = \frac{\underline{P}(q,e)}{\underline{P}(q,e) + \overline{P}(not\ q,e)}, \quad \overline{P}(q \mid e) = \frac{\overline{P}(q,e)}{\overline{P}(q,e) + \underline{P}(not\ q,e)}. \quad (2)$$

For the lower conditional probability $\underline{P}(q \mid e)$, if $\underline{P}(q,e) + \overline{P}(not\ q,e) = 0$ and $\overline{P}(q,e) > 0$, then $\underline{P}(q \mid e) = 1$. Similarly, for the upper conditional probability $\overline{P}(q \mid e)$, if $\overline{P}(q,e) + \underline{P}(not\ q,e) = 0$ and $\overline{P}(not\ q,e) > 0$, then $\overline{P}(q \mid e) = 0$. Both formulas are undefined if $\overline{P}(q,e)$ and $\overline{P}(not\ q,e)$ are 0.

To clarify, consider the following example.

Example 1.
The following PASP encodes a simple graph reachability problem.

```
0.2::edge(1,2).
0.3::edge(2,4).
0.9::edge(1,3).
path(X,Y):- connected(X,Z), path(Z,Y).
path(X,Y):- connected(X,Y).

connected(X,Y):- edge(X,Y), not nconnected(X,Y).
nconnected(X,Y):- edge(X,Y), not connected(X,Y).
```

The first three facts are probabilistic. The rules state that there is a path between X and Y if they are directly connected or if there is a path between Z and Y and X and Z are connected. Two nodes may or may not be connected if there is an edge between them. There are $2^3 = 8$ worlds to consider, listed in Table 1. If we want to compute the probability of the query $q = path(1,4)$, only w_6 and w_7 contribute to the upper bound (no contribution to the lower bound), obtaining $P(q) = [0, 0.06]$. If we observe

Table 1. *Worlds and their probabilities for Example 1. The second, third, and fourth columns contain 0 or 1 if the corresponding probabilistic fact is respectively false or true in the considered world. The LP/UP column indicates whether the considered world contributes to the lower (LP) or upper (UP) bound (or does not contribute, marked with a dash) for the probability of the query $path(1, 4)$*

w_{id}	$edge(1, 2)$	$edge(2, 4)$	$edge(1, 3)$	LP/UP	Probability
w_0	0	0	0	–	0.056
w_1	0	0	1	–	0.504
w_2	0	1	0	–	0.024
w_3	0	1	1	–	0.216
w_4	1	0	0	–	0.014
w_5	1	0	1	–	0.126
w_6	1	1	0	UP	0.006
w_7	1	1	1	UP	0.054

$e = edge(2, 4)$, we get $P(q, e) = [0, 0.06]$, $P(not\ q, e) = [0.24, 0.3]$, thus $\underline{P}(q \,|\, e) = 0$ and $\overline{P}(q \,|\, e) = 0.2$.

Inference in PASP can be expressed as a Second Level Algebraic Model Counting Problem (2AMC) (Kiesel et al., 2022). Given a propositional theory T where its variables are grouped into two disjoint sets, X_o and X_i, two commutative semirings $R^i = (D^i, \oplus^i, \otimes^i, n_{\oplus^i}, n_{\otimes^i})$ and $R^o = (D^o, \oplus^o, \otimes^o, n_{\oplus^o}, n_{\otimes^o})$, two weight functions, $w_i : lit(X_i) \to D^i$ and $w_o : lit(X_o) \to D^o$, and a transformation function $f : D^i \to D^o$, 2AMC is encoded as:

$$2AMC(T) = \bigoplus_{I_o \in \mu(X_o)}^{o} \bigotimes_{a \in I_o}^{o} w_o(a) \otimes^o f(\bigoplus_{I_i \in \varphi(T|I_o)}^{i} \bigotimes_{b \in I_i}^{i} w_i(b))$$

where $\varphi(T \,|\, I_o)$ is the set of assignments to the variables in X_i such that each assignment, together with I_o, satisfies T and $\mu(X_o)$ is the set of possible assignments to the variables in X_o. In other words, 2AMC requires to solve two Algebraic Model Counting (AMC) (Kimmig et al., 2017) tasks. The outer task focuses on the variables X_o, and for each possible assignment to these variables, an inner AMC task is performed by considering X_i. These two tasks are connected via a transformation function that turns values from the inner task into values for the outer task. Different instantiations of the components allow different tasks to be represented. To perform inference in PASP, Azzolini and Riguzzi (2023a) proposed to consider as innermost semiring $\mathcal{R}^i = (\mathbb{N}^2, +, \cdot, (0, 0), (1, 1))$ with X_i containing the atoms of the Herbrand base except the probabilistic facts and w_i mapping $not\ q$ to $(0, 1)$ and all other literals to $(1, 1)$, as outer semiring the two-dimensional probability semiring, that is, $\mathcal{R}^o = ([0, 1]^2, +, \cdot, (0, 0), (1, 1))$, with X_o containing the atoms of the probabilistic facts and w_o associating (p, p) and $(1 - p, 1 - p)$ to a and $not\ a$, respectively, for every probabilistic fact $p :: a$ and $(1, 1)$ to all the remaining literals, and as transformation function $f((n_1, n_2))$ returning the pair (v_{lp}, v_{up}) where $v_{lp} = 1$ if $n_1 = n_2$, 0 otherwise, and $v_{up} = 1$ if $n_1 > 0$, 0 otherwise.

2AMC can be solved via knowledge compilation (Darwiche and Marquis, 2002), often adopted in probabilistic logical settings, since it allows us to compactly represent the input theory with a tree or graph and then computing the probability of a query by traversing it. aspmc (Eiter et al., 2021, 2024) is one such tool, that has been proven more effective than other tools based on alternative techniques such as projected answer set enumeration (Azzolini et al., 2022; Azzolini and Riguzzi, 2023a). aspmc converts the input theory into a negation normal form (NNF) formula, a tree where each internal node is labeled with either a conjunction (and-node) or a disjunction (or-node), and leaves are associated with the literals of the theory. More precisely, aspmc targets sd-DNNFs which are NNFs with three additional properties: (i) the variables of children of and-nodes are disjoint (decomposability property); (ii) the conjunction of any pair of children of or-nodes is logically inconsistent (determinism property); and (iii) children of an or-node consider the same variables (smoothness property). Furthermore, aspmc also requires X-firstness (Kiesel et al., 2022), a property that constraints the order of appearance of variables: given two disjoint partitions X and Y of the variables in an NNF n, a node is termed pure if all variables departing from it are members of either X or Y. If this does not hold, the node is called mixed. An NNF has the X-firstness property if for each and-node, all of its children are pure nodes or if one child is mixed and all the other nodes are pure with variables belonging to X.

2.2 Parameter learning in probabilistic answer set programs

We adopt the same Learning from Interpretations framework of (Azzolini et al., 2024), that we recall here for clarity. We denote a PASP with $\mathcal{P}(\Pi)$, where Π is the set of parameters that should be learnt. The parameters are the probabilities associated to (a subset of) probabilistic facts. We call such facts as *learnable facts*. Note that the probabilities of some probabilistic facts can be fixed, that is there can be some probabilistic facts that are not learnable facts. A partial interpretation $I = \langle I^+, I^- \rangle$ is composed by two sets I^+ and I^- that respectively represent the set of true and false atoms. It is called partial since it may specify the truth value of some atoms only. Given a partial interpretation I, we call the interpretation query $q_I = \bigwedge_{i^+ \in I^+} i^+ \bigwedge_{i^- \in I^-} not\ i^-$. The probability of an interpretation I, $P(I)$, is defined as the probability of its interpretation query, which is associated with a probability range since we interpret the program under the CS. Given a PASP $\mathcal{P}(\Pi)$, the lower and upper probability for an interpretation I are defined as

$$\overline{P}(I \mid \mathcal{P}(\Pi)) = \sum_{w \in \mathcal{P}(\Pi) \ \mid \ \exists m \in AS(w),\ m \models I} P(w),$$

$$\underline{P}(I \mid \mathcal{P}(\Pi)) = \sum_{w \in \mathcal{P}(\Pi) \ \mid \ \forall m \in AS(w),\ m \models I} P(w).$$

Definition 1 (Parameter Learning in probabilistic answer set programs)
Given a PASP $\mathcal{P}(\Pi)$ and a set of (partial) interpretations $\mathcal{I}$, the goal of the parameter learning task is to find a probability assignment to the probabilistic facts such that the

product of the lower (or upper) probabilities of the partial interpretations is maximized, that is solve:

$$\Pi^* = \arg\max_\Pi \underline{P}(\mathcal{I} \mid \mathcal{P}(\Pi)) = \arg\max_\Pi \prod_{I \in \mathcal{I}} \underline{P}(I \mid \mathcal{P}(\Pi))$$

which can be equivalently expressed as

$$\Pi^* = \arg\max_\Pi \log(\underline{P}(\mathcal{I} \mid \mathcal{P}(\Pi))) = \arg\max_\Pi \sum_{I \in \mathcal{I}} \log(\underline{P}(I \mid \mathcal{P}(\Pi))) \tag{3}$$

also known as *log-likelihood* (LL). The maximum value of the LL is 0, obtained when all the interpretations have probability 1. The use of log probabilities is often preferred since summations instead of products are considered, thus possibly preventing numerical issues, especially when many terms are close to 0.

Note that, since the probability of a query (interpretation) is described by a range, we need to select whether we maximize the lower or upper probability. A solution that maximizes one of the two bounds may not be a solution that also maximizes the other bound. To see this, consider the program $\{\{q :\!- a, b, not\ nq\}, \{nq :\!- a, b, not\ q\}\}$ where a and b are both probabilistic with probability p_a and p_b, respectively. Suppose we have the interpretation $I = \langle \{q\}, \{\} \rangle$. Here, $\underline{P}(I) = 0$ while $\overline{P}(I) = p_a \cdot p_b$. Thus, any probability assignment to p_a and p_b maximizes the lower probability (which is always 0) but only the assignment $p_a = 1$ and $p_b = 1$ maximizes $\overline{P}(I)$.

Example 2.
Consider the program shown in Example 1. Suppose we have two interpretations: $I_0 = \langle \{path(1,3)\}, \{path(1,4)\} \rangle$ and $I_1 = \langle \{path(1,4)\}, \{\} \rangle$. Thus, we have two interpretation queries: $q_{I_0} = path(1,3),\ not\ path(1,4)$ and $q_{I_1} = path(1,4)$. Suppose that the probabilities of all the four probabilistic facts can be set and call this set Π. The parameter learning task of Definition 1 involves solving: $\Pi^* = \arg\max_\Pi(\log(P(q_{I_0} \mid \Pi)) + \log(P(q_{I_1} \mid \Pi)))$.

Azzolini et al. (2024) focused on ground probabilistic facts whose probabilities should be learnt and proposed an algorithm based on Expectation Maximization (EM) to solve the task. Suppose that the target is the upper probability. The treatment for the lower probability is analogous and only differs in the considered bound. The EM algorithm alternates an expectation phase and a maximization phase, until a certain criterion is met (usually, the difference between two consecutive iterations is less than a given threshold). This involves computing, in the expectation phase, for each probabilistic fact a_i whose probability should be learnt:

$$\overline{E}[a_{i0}] = \sum_{I \in \mathcal{I}} \overline{P}(not\ a_i \mid I), \quad \overline{E}[a_{i1}] = \sum_{I \in \mathcal{I}} \overline{P}(a_i \mid I). \tag{4}$$

These values are used in the maximization step to update each parameter Π_i as:

$$\Pi_i = \frac{\overline{E}[a_{i1}]}{\overline{E}[a_{i0}] + \overline{E}[a_{i1}]} = \frac{\sum_{I \in \mathcal{I}} \overline{P}(a_i \mid I)}{\sum_{I \in \mathcal{I}} \overline{P}(not\ a_i \mid I) + \overline{P}(a_i \mid I)}. \tag{5}$$

3 Algorithms for parameter learning

We propose two algorithms for solving the parameter learning task, both based on the extraction of symbolic equations from the NNF (Darwiche and Marquis, 2002) representing a query. So, we first describe this common part. In the following, when we consider the probability of a query we focus on the upper probability. The treatment for the lower probability is analogous and only differs in the considered bound.

3.1 Extracting equations from a NNF

The upper probability of a query q is computed as a sum of products (see equation 1). If, instead of using the probabilities of the facts, we keep them symbolic (i.e. with their name), we can extract a nonlinear symbolic equation for the query, where the variables are the parameters associated with the learnable facts. Call this equation $f_{up}(\Pi)$ where Π is the set of parameters. Its general form is $f_{up}(\Pi) = \sum_{w_i} \prod_{a_j \in w_i} p_j \prod_{a_j \notin w_i} (1 - p_j) \cdot k_i$ where k_i is the contribution of the probabilistic facts with fixed probability for world w_i. We can cast the task of extracting an equation for a query as a 2AMC problem. To do so, we can consider as inner semiring and as transformation function the ones proposed by Azzolini and Riguzzi (2023a) and described in Section 2.1. From this inner semiring we obtain two values, one for the lower and one for the upper probability. The sensitivity semiring by Kimmig et al. (2017) allows the extraction of an equation from an AMC task. We have two values to consider, so we extend that semiring to $R^o = (\mathbb{R}[X], +, -, (0, 0), (1, 1))$ with

$$
w_o(l) = \begin{cases}
(p, p) & \text{for a p.f. } p :: a \text{ with fixed probability and } l = a \\
(1 - p, 1 - p) & \text{for a p.f. } p :: a \text{ with fixed probability and } l = not\ a \\
(\pi_a, \pi_a) & \text{for a learnable fact } \pi_a :: a \text{ and } l = a \\
(1 - \pi_a, 1 - \pi_a) & \text{for a learnable fact } \pi_a :: a \text{ and } l = not\ a \\
(1, 1) & \text{otherwise}
\end{cases}
$$

where p.f. stands for probabilistic fact and $\mathbb{R}[X]$ is the set of real valued functions parameterized by X. Variable π_a indicates the symbolic probability of the learnable fact a. In this way, we obtain a nonlinear equation that represents the probability of a query. When evaluated by replacing variables with actual numerical values, we obtain the probability of the query when the learnable facts have such values. Simplifying the obtained equation is also a crucial step since it might significantly reduce the number of operations needed to evaluate it.

Example 3
If we consider Example 1 with query $path(1, 4)$ and associate a variable π_{xy} with each $edge(x, y)$ probabilistic fact (and consider them as learnable), the symbolic equation for its upper probability, that is f_{up}, is $\pi_{12} \cdot \pi_{24} \cdot \pi_{13} + \pi_{12} \cdot \pi_{24} \cdot (1 - \pi_{13})$, that can be simplified to $\pi_{12} \cdot \pi_{24}$.

We now show how to adopt symbolic equations to solve the parameter learning task.

Algorithm 1 Function LEARNINGOPT: solving the parameter learning task targeting the upper probability with constrained optimization in a PASP P with learnable probabilistic facts Π and interpretations I.

1: **function** LEARNINGOPT(P, I)
2:     $eqsList \leftarrow \emptyset$
3:     **for** $i \in I$ **do**
4:         $eq \leftarrow$ GETEQUATIONFROMNNF(i)
5:         $eqsList \leftarrow eqsList \cup$ SIMPLIFY(eq)
6:     **end for**
7:     $\Pi^* \leftarrow$ SOLVEOPTIMIZATIONPROBLEM($eqsList$)
8:     **return** Π^*
9: **end function**

3.2 Solving parameter learning with constrained optimization

The learning task requires maximizing the sum of the log probabilities of the interpretations equation (3) where the tunable parameters are the facts whose probability should be learnt. Algorithm 1 sketches the involved steps. We can extract the equation for the probability of each interpretation from the NNF and consider each parameter of a learnable fact as a variable (function GETEQUATIONFROMNNF). To reduce the size of the equation we simplify it (function SIMPLIFY). Then, we set up a constrained nonlinear optimization problem where the target is the maximization of the sum of the equations representing the log probabilities of the interpretations (function SOLVEOPTIMIZATIONPROBLEM). We need to also impose that the parameters of the learnable facts are between 0 and 1. In this way, we can easily adopt off-the-shelf solvers and do not need to write a specialized one.

3.3 Solving parameter learning with expectation maximization

We propose another algorithm that instead performs Expectation Maximization as per equation (4) and (5). It is sketched in Algorithm 2. For each interpretation I, we add its interpretation query q_I into the program. To compute $\overline{P}(a_j \mid I_k)$ for each learnable probabilistic fact a_j and each interpretation I_k we proceed as follows: first, we compute $\overline{P}(a_j, I_k)$ and $\underline{P}(not\ a_j, I_k)$. Then, equation (2) allows us to compute $\overline{P}(a_j \mid I_k)$. Similarly for $\underline{P}(a_j \mid I_k)$. We extract an equation for each of these queries (function GETEQUATIONSFROMNNF) and iteratively evaluate them until the convergence of the EM algorithm (lines 5–10 that alternates the expectation phase with function EXPECTATION, the maximization phase with function MAXIMIZATION, and the computation of the log-likelihood with function COMPUTELL). We consider as default convergence criterion a variation of the log-likelihood less than $5 \cdot 10^{-4}$ between two subsequent iterations. However, this parameter can be set by the user. If we denote with n_p the number of probabilistic facts whose probabilities should be learnt and n_i the number of interpretations, we need to extract equations for $2 \cdot n_p \cdot n_i$ queries. However, this is possible

---

Algorithm 2 Function LEARNINGEM: solving the parameter learning task targeting the upper probability with Expectation Maximization in a PASP P with learnable probabilistic facts Π and with interpretations I.

1: **function** LEARNINGEM(P, I)
2:     $eqsList \leftarrow$ GETEQUATIONSFROMNNF(π, I)
3:     $ll_0 \leftarrow 0$
4:     $ll_1 \leftarrow 1$
5:     **while** $(ll_1 - ll_0) > 5 \cdot 10^{-4}$ **do**   ▷ Loop until convergence.
6:         $ll_0 \leftarrow ll_1$
7:         $E \leftarrow$ EXPECTATION($eqsList$)
8:         $\Pi \leftarrow$ MAXIMIZATION(E)
9:         $ll_1 \leftarrow$ COMPUTELL($eqsList, \Pi$)
10:     **end while**
11:     **return** Π
12: **end function**

---

with only one pass of the NNF: across different iterations, the structure of the program is the same, only the probabilities, and thus the result of the queries, will change. Thus, we can store the performed operations in memory and use only those to reevaluate the probabilities, without having to rebuild the NNF at each iteration.

Example 4

Consider Example 2 and its two interpretation queries, q_{I_0} and q_{I_1}. If we consider their symbolic equations we have $\pi_{12} \cdot \pi_{24}$ (see Example 3) and $\pi_{13} \cdot (\pi_{12} + \pi_{24} - \pi_{12} \cdot \pi_{24})$, where with π_{xy} we indicate the probability of $edge(x, y)$. If we compactly denote with Π the set of all the probabilities (i.e. all the π_{xy}), the optimization problem of equation (3) requires solving $\Pi^* = \arg \max_{\Pi}(\log(\pi_{12} \cdot \pi_{24}) + \log(\pi_{13} \cdot (\pi_{12} + \pi_{24} - \pi_{12} \cdot \pi_{24})))$. The optimal solution is to set the values of all the π_{xy} to 1, obtaining a log-likelihood of 0. Note that, in general, it is not always possible to obtain a LL of 0.

4 Related work

There are many different techniques available to solve the parameter learning task, but most of them only work for programs with a unique model per world: PRISM (Sato, 1995) was one of the first tools considering inference and parameter learning in PLP. Its first implementation, which dates back to 1995, offered an algorithm based on EM. The same approach is also adopted in EMBLEM (Bellodi and Riguzzi, 2013), which learns the parameters of Logic Programs with Annotated Disjunctions (LPADs) (Vennekens et al., 2004), that is logic programs with disjunctive rules where each head atom is associated with a probability, and in ProbLog2 (Fierens et al., 2015) that learn the parameters of ProbLog programs from partial interpretations adopting the LFI-ProbLog algorithm of Gutmann et al. (2011). LeProbLog Gutmann et al. (2008) still considers ProbLog programs but uses gradient descent to solve the task.

Few tools consider PASP and allow multiple models per world. dPASP (Geh et al., 2023) is a framework to perform parameter learning in Probabilistic Answer Set Programs

but targets the max-ent semantics, where the probability of a query is the sum of the probabilities of the models where the query is true. They propose different algorithms based on the computation of a fixed point and gradient ascent. However, they do not target the Credal Semantics. Parameter Learning under the Credal Semantics is also available in PASTA (Azzolini et al., 2022, 2024). Here, we adopt the same setting (learning from interpretations) but we address the task with an inference algorithm based on 2AMC and extraction of symbolic equations, rather than projected answer set enumeration, which has been empirically proven more effective (Azzolini and Riguzzi, 2023a). This is also proved in our experimental evaluation.

Lastly, there are alternative semantics to adopt in the context of Probabilistic Answer Set Programming, namely, P-log (Baral et al., 2009), LPMLN (Lee and Wang, 2016), and smProbLog (Totis et al., 2023). The relation among these has been partially explored by (Lee and Yang, 2017) but a complete treatment providing a general comparison is still missing. The parameter learning task under these semantics has been partially addressed and an in-depth comparison between all the existing approaches can be an interesting future work.

5 Experiments

We ran the experiments on a computer with 16 GB of RAM running at 2.40 GHz with 8 hours of time limit. The goal of the experiments is many-fold: (i) finding which one of the algorithms is faster; (ii) discovering which one of the algorithms better solves the optimization problem represented by equation (3) (note that the sum of the log probabilities is maximum at 0, and this happens when all the partial interpretations have probability 1); (iii) evaluating whether different initial probability values impact on the execution time; and (iv) comparing our approach against the algorithm based on projected answer set enumeration of Azzolini et al. (2024) called PASTA. We considered only PASTA since it is the only algorithm that currently solves the task of parameter learning in PASP under the credal semantics. We used aspmc (Eiter et al., 2021) as backend solver for the computation of the formula, SciPy version 1.13.0 as optimization solver (Virtanen et al., 2020), and SymPy (Meurer et al., 2017) version 1.12 to simplify equations. In this way, the tool is completely open source.[1] In the experiments, for the algorithm based on constrained optimization (Section 3.2) we tested two nonlinear optimization algorithms available in SciPy, namely COBYLA (Powell, 1994) and SLSQP (Kraft, 1994). COBYLA stands for Constrained Optimization BY Linear Approximation and is a derivative-free nonlinear constrained optimization algorithm based on linear approximation of the objective function and constraints during the solving process. On the contrary, SLSQP is based on Sequential Least Squares Programming, which is based on solving, at each iteration, a least square problem equivalent to the original problem. In the results, we denote the algorithm based on Expectation Maximization described in Section 3.3 with EM.

---

[1] Datasets and source code available at https://github.com/damianoazzolini/aspmc and on Zenodo with DOI 10.5281/zenodo.12667046.

5.1 Datasets descriptions

We considered four datasets. Where not explicitly specified, all the initial probabilities of the learnable facts are set to 0.5. For all the instances, we generated configurations with 5, 10, 15, and 20 interpretations. The atoms to be included in the interpretations are taken uniformly at random from the Herbrand base of each program and some of them are considered as negated, again uniformly at random.

The *coloring* dataset models the graph coloring problem, where each node is associated with a color and configurations are considered valid only if nodes associated with the same color are not connected. We do not explicitly remove invalid solutions but only mark them as not valid. This makes the following dataset crucial to test our proposal when there are an increasing number of answer sets for each world. All the instances have the following rules:

```
red(X) :- node(X), not green(X), not blue(X).
green(X) :- node(X), not red(X), not blue(X).
blue(X) :- node(X), not red(X), not green(X).
e(X,Y) :- edge(X,Y).
e(Y,X) :- edge(Y,X).
c0 :- e(X,Y), red(X), red(Y).
c1 :- e(X,Y), green(X), green(Y).
c2 :- e(X,Y), blue(X), blue(Y).
valid :- not c0, not c1, not c2.
```

The goal is to learn the probabilities of the *edge*/2 facts given an increasing number of interpretations containing the observed colors for some edges and whether the configuration is *valid* or not. We considered complete graphs of size 4 (*coloring4*) and 5 (*coloring5*). Here the interpretations have a random length between 3 and 4.

The *path* dataset models a path reachability problem and every instance has the rules described in Example 1. We considered this dataset since it represents a graph structure that can model many problems. The goal is to learn the probabilities of the *edge*/2 facts given that some paths are observed. We considered random connected graphs with 10 (*path10*) and 15 (*path15*) edges. Here the interpretations have a random length between 1 and 3.

The *shop* dataset models the shopping behavior of some people, with different products available where some of them cannot be bought together. An example of instance (with 2 people) is:

```
bought(spaghetti,john):-shops(john), not bought(steak,john).
bought(steak,john):-shops(john), not bought(spaghetti,john).
bought(spaghetti,mary):-shops(mary), not bought(beans,mary).
bought(beans,mary):-shops(mary), not bought(spaghetti,mary).
bought(spaghetti):-  bought(spaghetti,_).
bought(steak):- bought(steak,_).
:- bought(spaghetti), bought(steak).
```

The goal is to learn the probabilities of the *shops*/1 learnable facts given an increasing number of products that are observed being bought (or not bought) and an increasing number of people. We considered instances with 4 (*shop4*), 8 (*shop8*), 10 (*shop10*), and 12 (*shop12*) people. Here the interpretations have a random length between 1 and 10. With this dataset we assess the learning algorithms in programs when constraints prune some solutions.

The *smoke* dataset, adapted from (Totis et al., 2023), models a network where some people smoke and others are influenced by the smoking behavior of their friends. This is a well-known dataset often used to benchmark probabilistic logic programming systems. An example of instance with two people is:

```
0.1::asthma_f(1). 0.4::asthma_fact(1). 0.3::stress(1).
0.1::asthma_f(2). 0.4::asthma_fact(2). 0.3::stress(2).
0.2 :: predisposition.

smokes(X) :- stress(X).
smokes(X) :- influences(Y, X), smokes(Y).
asthma_rule(X):- smokes(X), asthma_fact(X).

asthma(X) :- asthma_f(X).
asthma(X) :- asthma_rule(X).

ill(X)  :- smokes(X), asthma(X), not n_ill(X).
n_ill(X):- smokes(X), asthma(X), predisposition, not ill(X).
```

Here, the learnable facts have signature *influences*/2 and the goal is to learn their probabilities given that some *ill*/1 facts are observed. We consider instances with 3 (*smoke2*), 4 (*smoke4*), and 6 (*smoke6*) people. Here the interpretations have a random length between 1 and 3.

5.2 Results

First, we compare the execution times of the four proposed algorithms. Results are reported in Figure 1. COBYLA and SLSQP have comparable execution times while EM and PASTA are often the slowest. The only instances where PASTA is faster are *coloring4* and *coloring5*. Given the imposed time and memory limits, EM cannot solve the instances *shop12* with any number of interpretations and *shop10* with 15 and 20 interpretations, and *shop8* with 20 interpretations, *coloring5* with any number of interpretations, and *path15* with any number of interpretations (all due to memory limit) while PASTA was not able to solve *smoke6* with any number of interpretations and *smoke5* with 10, 15, and 20 interpretations (all due to time limit). The two algorithms based on constrained optimization are able to solve every considered instance.
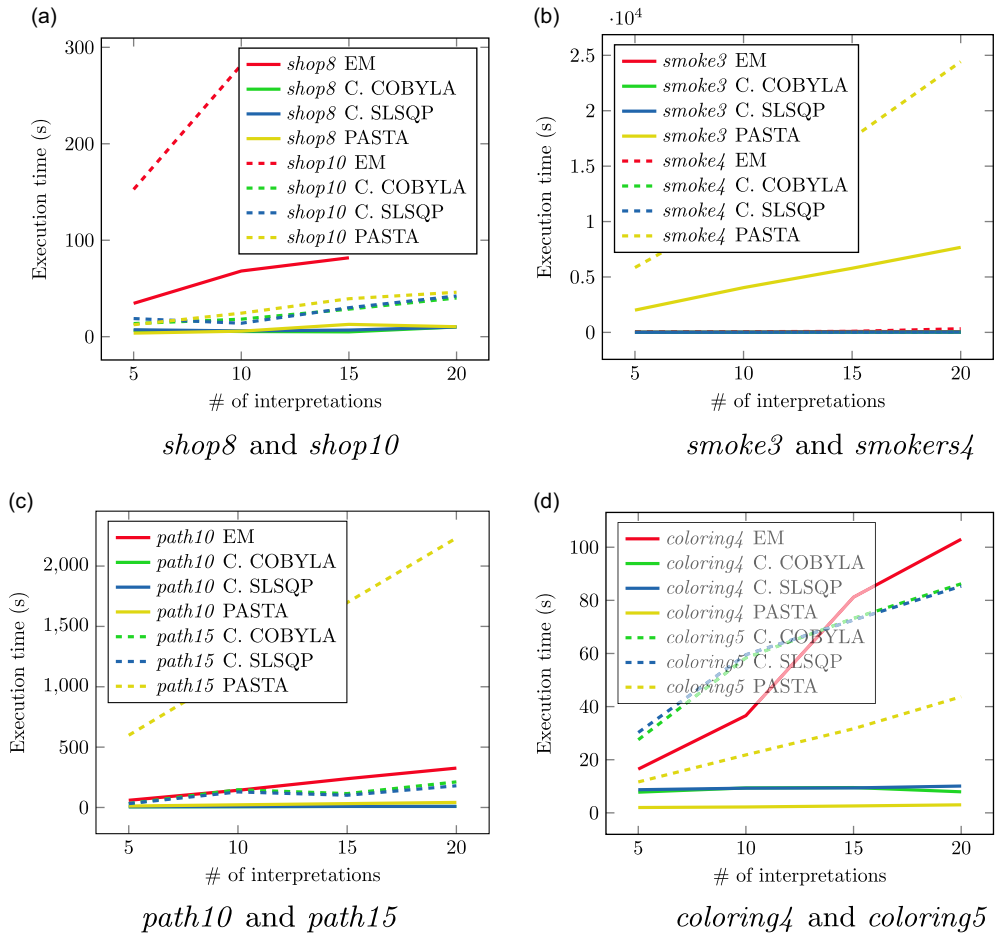
Fig. 1. Execution times for EM, constrained optimization solved with COBYLA and SLSQP, and PASTA, by increasing the number of interpretations. For *path15* and *coloring5* the line for EM is missing since it cannot solve any instance. The initial probabilities for learnable facts are set to 0.5.

We also evaluated the algorithms in terms of final log-likelihood. Table 2 shows the results on the *coloring4*, *path10*, *shop4*, *shop8*, *smoke3*, and *smoke4* instances. SLSQP has the best performance: it seems to be able to better maximize the LL in *coloring4*, *shop4*, and *shop8*, while COBYLA cannot reach such maxima. For the other two, the results are equal. EM is also competitive with SLSQP. PASTA has the worst performance for all the datasets and cannot reach a LL of 0 while other algorithms, such as the ones based on constrained optimization, succeed.

Lastly, we also run experiments with the two optimization algorithms by considering different initial probability values. Figure 2 shows the results. In general, the initial probability values have little influence on the execution time for SLSQP. The impact is more evident on COBYLA: for example, for *coloring5* with 10 interpretations, setting the initial values to 0.1 requires 100 s of computation while setting them to 0.5

Table 2. *Final log-likelihood (LL) values for the tested algorithms on six selected instances with the initial probability of the learnable facts set to 0.5. The column # int. contains the number of interpretations considered, the column EM contains the results obtained with expectation maximization, columns C. COBYLA and C. SLSQP stands for constrained optimization solved with, respectively, COBYLA and SLSQP, and the column PASTA contains the results obtained with the PASTA solver*

	coloring4				shop8			
# int.	EM	C. COBYLA	C. SLSQP	PASTA	EM	C. COBYLA	C. SLSQP	PASTA
5	-0.016	0.000	0.000	-34.539	-1.389	-12.710	-1.385	-34.077
10	-0.049	0.000	0.000	-55.262	-2.257	0.000	0.000	-63.268
15	-0.044	0.000	0.000	-96.709	-1.923	0.000	-1.902	-75.986
20	-0.030	-36.775	0.000	-124.340	-1.388	-72.524	-0.011	-127.679

	path10				smoke4			
# int.	EM	C. COBYLA	C. SLSQP	PASTA	EM	C. COBYLA	C. SLSQP	PASTA
5	-0.002	0.000	0.000	-27.631	-9.430	-9.423	-9.423	-20.723
10	-0.004	0.000	0.000	-48.354	-25.164	-25.160	-25.160	-48.354
15	-0.007	0.000	0.000	-75.985	-50.023	-50.020	-50.020	-96.709
20	-0.004	0.000	0.000	-103.616	-64.576	-64.571	-64.571	-110.524

	smoke3				shop4			
# int.	EM	C. COBYLA	C. SLSQP	PASTA	EM	C. COBYLA	C. SLSQP	PASTA
5	-14.633	-14.630	-14.630	-27.631	-1.462	-150.606	0.000	-20.633
10	-36.711	-36.708	-36.708	-69.078	-5.669	0.000	0.000	-54.992
15	-35.172	-35.166	-35.166	-69.078	-7.283	-49.346	0.000	-74.710
20	-72.872	-72.869	-72.869	-110.524	-2.770	0.000	-2.768	-101.824

or 0.9 requires 60 s. We extended this evaluation also to EM, but we do not report the results since the initial value makes no difference in terms of execution time. In the current version, PASTA does not allow setting an initial probability value different from 0.5.

Overall, the algorithm based on constrained optimization outperforms in terms of efficiency and in terms of final log-likelihood both the algorithm based on EM and PASTA. EM also outperforms PASTA in terms of final LL but it often requires excessive memory and cannot solve instances solvable by PASTA. Nevertheless, several improvements may be integrated within our approach to possibly increase the scalability: (i) considering different alternatives for the representation of symbolic equations, possibly more compact, also considering symmetries (Azzolini and Riguzzi, 2023b); (ii) developing ad-hoc simplification algorithms to reduce the size of the symbolic equations since they are only composed by summations of products. Furthermore, considering higher runtimes could be beneficial for enumeration-based techniques.
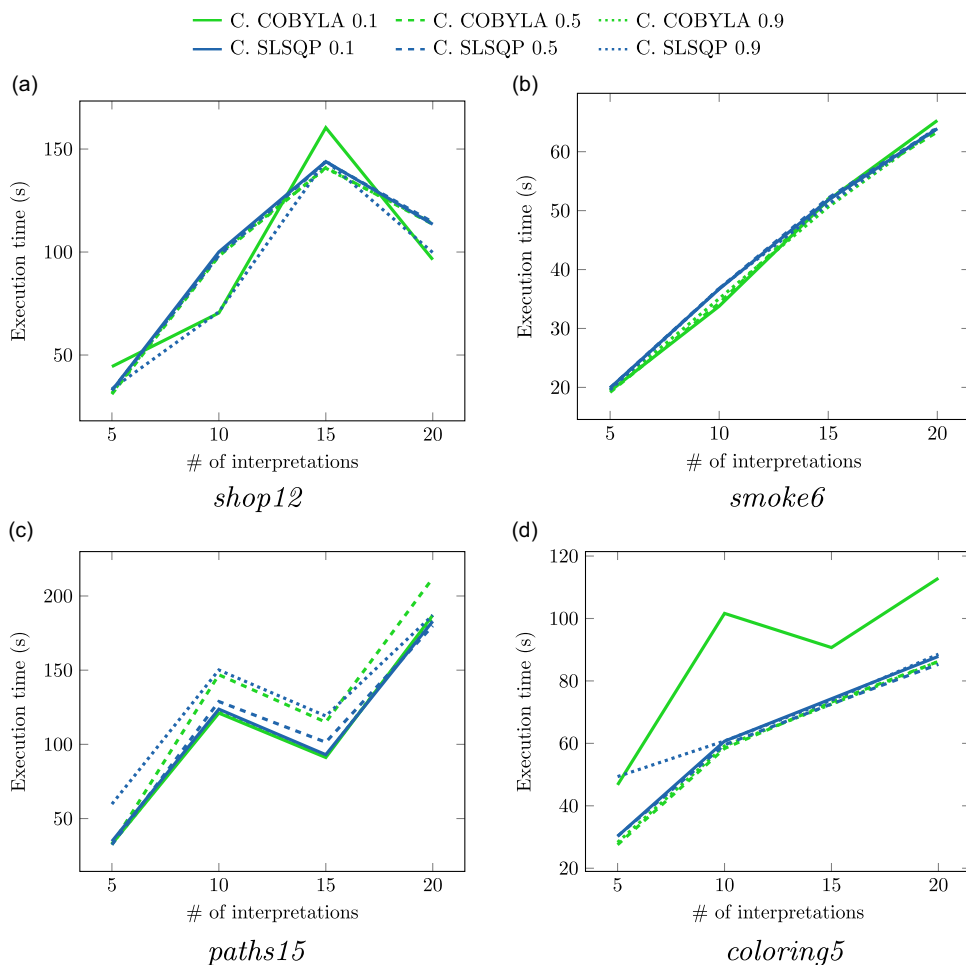
Fig. 2. Execution times of constrained optimization solved with COBYLA and SLSQP on *coloring5*, *path15*, *shop12*, and *smoke6* with 0.1, 0.5, and 0.9 as initial values for the learnable facts.

6 Conclusions

In this paper, we propose two algorithms to solve the task of parameter learning in PASP. Both are based on the extraction of a symbolic equation from a compact representation of the problem but differ in the solving approach: one is based on Expectation Maximization while the other is based on constrained optimization. For the former, we tested two algorithms, namely COBYLA and SLSQP. We compare them against PASTA, a solver adopting projected answer set enumeration. Empirical results show that the algorithms based on constrained optimization are more accurate and also faster than the ones based on EM and PASTA. Furthermore, the one based on EM is still competitive with PASTA but often requires an excessive amount of memory. As a future work we plan to consider the parameter learning task for other semantics and also theoretically study

the complexity of the task in light of existing complexity results for inference (Mauá and Cozman, 2020).

Acknowledgements

Competing interests

The authors declare none.

References

AZZOLINI, D. 2023. A constrained optimization approach to set the parameters of probabilistic answer set programs. In *Inductive Logic Programming*, E. Bellodi, F. A. Lisi and R. Zese, Eds. Springer Nature Switzerland, Cham, 1–15.

AZZOLINI, D., BELLODI, E. and RIGUZZI, F. 2022. Statistical statements in probabilistic logic programming . In *Logic Programming and Nonmonotonic Reasoning*, G. Gottlob, D. Inclezan and M. Maratea, Eds., Springer International Publishing, Cham, 43–55.

AZZOLINI, D., BELLODI, E. and RIGUZZI, F. 2024. Learning the parameters of probabilistic answer set programs. In *Inductive Logic Programming*, S. H. Muggleton and A. Tamaddoni-Nezhad, Eds. Springer Nature Switzerland, Cham, 1–14.

AZZOLINI, D. and RIGUZZI, F. 2021. Optimizing probabilities in probabilistic logic programs. *Theory and Practice of Logic Programming* 21, 5, 543–556.

AZZOLINI, D. and RIGUZZI, F. 2023a. Inference in probabilistic answer set programming under the credal semantics. In *AIxIA. 2023 - Advances in Artificial Intelligence*, BASILI, R., LEMBO, D., LIMONGELLI, C. and ORLANDINI, A., Eds. 14318. Heidelberg, Germany, Springer, 367–380. *Lecture Notes in Artificial Intelligence*.

AZZOLINI, D. and RIGUZZI, F. 2023b. Lifted inference for statistical statements in probabilistic answer set programming. *International Journal of Approximate Reasoning* 163, 109040.

BARAL, C., GELFOND, M. and RUSHTON, N. 2009. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming* 9, 1, 57–144.

BELLODI, E. and RIGUZZI, F. 2013. Expectation maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis* 17, 2, 343–363.

COZMAN, F. G. and MAUÁ, D. D. 2016. The structure and complexity of credal semantics. In HOMMERSOM, A. and ABDALLAH, S. A., Eds. 3rd International Workshop on Probabilistic Logic Programming (PLP 2016), volume 1661 of CEUR Workshop Proceedings, 3–14.CEUR-WS.org.

COZMAN, F. G. and MAUÁ, D. D. 2020. The joy of probabilistic answer set programming: Semantics, complexity, expressivity, inference. *International Journal of Approximate Reasoning* 125, 218–239.

DARWICHE, A. and MARQUIS, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17, 229–264.

DE RAEDT, L., KIMMIG, A. and TOIVONEN, H. 2007.ProbLog: A probabilistic Prolog and its application in link discovery. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, VELOSO, M. M., ed.7, AAAI Press,2462–2467.

DRIES, A., KIMMIG, A., MEERT, W., RENKENS, J., VAN DEN BROECK, G., VLASSELAER, J. and DE RAEDT, L. 2015. ProbLog2: Probabilistic logic programming. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2015), Springer, 9286, 312–315.Lecture Notes in Computer Science.

EITER, T., HECHER, M. and KIESEL, R. 2021. Treewidth-aware cycle breaking for algebraic answer set counting. *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR* 2021, 269–279.

EITER, T., HECHER, M. and KIESEL, R. 2024. Aspmc: New frontiers of algebraic answer set counting. *Artificial Intelligence* 330, 104109.

FIERENS, D., VAN DEN BROECK, G., RENKENS, J., SHTERIONOV, D. S., GUTMANN, B., THON, I., JANSSENS, G. and DE RAEDT, L. 2015. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming* 15, 3, 358–401.

GEH, R. L., GONCALVES, J., SILVEIRA, I. C., MAUA, D. D. and COZMAN, F. G. 2023. DPASP: A comprehensive differentiable probabilistic answer set programming environment for neurosymbolic learning and reasoning, *CoRR*. https://doi.org/10.48550/arXiv.2308.02944

GELFOND, M. and LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In 5th International Conference and Symposium on Logic Programming (ICLP/SLP), MIT Press, USA, 88, 1070–1080.

GUTMANN, B., KIMMIG, A., KERSTING, K. and RAEDT, L. D. 2008. Parameter learning in probabilistic databases: A least squares approach. Lecture Notes in Computer Science. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2008), Springer, 5211, 473–488.

GUTMANN, B., THON, I. and DE RAEDT, L. 2011. Learning the parameters of probabilistic logic programs from interpretations. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2011)*, GUNOPULOS, D., HOFMANN, T., MALERBA, D. and VAZIRGIANNIS, M. Eds.6911, Springer, 581–596. Lecture Notes in Computer Science.

KIESEL, R., TOTIS, P. and KIMMIG, A. 2022. Efficient knowledge compilation beyond weighted model counting. *Theory and Practice of Logic Programming* 22, 4, 505–522.

KIMMIG, A., VAN DEN BROECK, G. and DE RAEDT, L. 2017. Algebraic model counting. *Journal of Applied Logic* 22, C, 46–62.

KRAFT, D. 1994. Algorithm 733: TOMP-fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software* 20, 3, 262–281.

LEE, J. and WANG, Y. 2016. Weighted rules under the stable model semantics. In BARAL, C., DELGRANDE, J. P. and WOLTER, F., Eds. Proc. of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 145–154.

LEE, J. and YANG, Z. 2017. LPMLN, weak constraints, and P-log. In SINGH, S. and MARKOVITCH, S., Proc. of the Thirty-First AAAI Conference on Artificial Intelligence, 4-9 Feb. 2017, AAAI Press, San Francisco, California, USA,1170–1177.

LUKASIEWICZ, T. 2005. Probabilistic description logic programs. In GODO, L., Ed. Lecture Notes in Computer Science, Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU, 6-8 July, 2005, Springer, Barcelona, Spain, 3571, 737–749.

MAUÁ, D. D. and COZMAN, F. G. 2020. Complexity results for probabilistic answer set programming. *International Journal of Approximate Reasoning* 118, 133–154.

MEURER, A., SMITH, C. P., PAPROCKI, M., ČERTÍK, O., KIRPICHEV, S. B., ROCKLIN, M., KUMAR, A., IVANOV, S., MOORE, J. K., SINGH, S., RATHNAYAKE, T., VIG, S., GRANGER, B. E., MULLER, R. P., BONAZZI, F., GUPTA, H., VATS, S., JOHANSSON, F., PEDREGOSA, F., CURRY, M. J., TERREL, A. R., ROUČKA, v., SABOO, A., FERNANDO, I., KULAL, S., CIMRMAN, R. and SCOPATZ, A. 2017. SymPy: symbolic computing in python. *PeerJ Computer Science* 3, e103.

POWELL, M. J. D. 1994. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*,GOMEZ, S. and HENNART, J.-P. Eds. Dordrecht,Springer Netherlands, 51–67.

RAEDT, L. D., KERSTING, K., NATARAJAN, S. and POOLE, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10, 2, 1–189.

RICHARDSON, M. and DOMINGOS, P. 2006. Markov logic networks. *Machine Learning* 62, 1, 107–136.

RIGUZZI, F. 2022. *Foundations of Probabilistic Logic Programming Languages, Semantics, Inference and Learning.* 2nded. River Publishers, Gistrup, Denmark.

SATO, T. 1995. A statistical learning method for logic programs with distribution semantics. In STERLING, L., Ed. Logic Programming, Proeedings of the Twelfth International Conference on Logic Programming, 13-16 June 1995, MIT Press, Tokyo, Japan,715–729.

TOTIS, P., DE RAEDT, L. and KIMMIG, A. 2023. smProbLog: Stable model semantics in ProbLog for probabilistic argumentation, *Theory and Practice of Logic Programming*, 1–50. https://doi.org/10.1017/S147106842300008X

VENNEKENS, J., VERBAETEN, S. and BRUYNOOGHE, M. 2004.Logic programs with annotated disjunctions. In *20th International Conference on Logic Programming (ICLP 2004)*, DEMOEN, B. and LIFSCHITZ, V., Eds. 3131. Berlin, Heidelberg,Springer, 431–445, Lecture Notes in Computer Science.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W., BRIGHT, J., van der WALT, S. J., BRETT, M., WILSON, J., MILLMAN, K. J., MAYOROV, N., NELSON, A. R. J., JONES, E., KERN, R., LARSON, E., CAREY, C. J., POLAT, İ., FENG, Y., MOORE, E. W., VANDERPLAS, J., LAXALDE, D., PERKTOLD, J., CIMRMAN, R., HENRIKSEN, I., QUINTERO, E. A., HARRIS, C. R., ARCHIBALD, A. M., RIBEIRO, A. H.,PEDREGOSA, F.,van MULBREGTP. and SciPy 1.0 Contributors 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods* 17, 3, 261–272.