Most Probable Explanation in Probabilistic Answer Set Programming

Damiano Azzolini[1] , **Giuseppe Mazzotta**[2] , **Francesco Ricca**[2] and **Fabrizio Riguzzi**[1]
[1]University of Ferrara, Ferrara, Italy
[2]University of Calabria, Rende, Italy
{damiano.azzolini, fabrizio.riguzzi}@unife.it, {giuseppe.mazzotta, francesco.ricca}@unical.it

Abstract

Most Probable Explanation (MPE) is a fundamental problem in statistical relational artificial intelligence. In the context of Probabilistic Answer Set Programming (PASP), solving MPE is still an open research problem. In this paper, we present three novel approaches for solving the MPE task in PASP that are based on: i) Algebraic Model Counting, ii) Answer Set Programming (ASP), and iii) ASP with quantifiers (ASP(Q)). These approaches are implemented and evaluated against existing solvers across different datasets and configurations. Empirical results demonstrate that the novel solutions consistently outperform existing alternatives for non-stratified programs.

1 Introduction

Statistical Relational Artificial Intelligence (StarAI) [Raedt *et al.*, 2016] serves as umbrella term encompassing a wide range of approaches that integrate probability into logic-based languages. These approaches have recently gained renewed momentum thanks to the advent of the field of neuro-symbolic integration [d'Avila Garcez *et al.*, 2019]. Here, we focus on the Probabilistic Answer Set Programming (PASP) formalism [Cozman and Mauá, 2020; Mauá and Cozman, 2020] and on the fundamental task of Most Probable Explanation (MPE) [Raedt *et al.*, 2016], which, interestingly, is still an open problem for PASP [Azzolini *et al.*, 2022]. Given a probabilistic answer set program and evidence e (in the form of conjunction of ground literals), MPE requires computing the world which has the highest probability and where e is cautiously or bravely true. These states are called lower and upper MPE states, respectively.

In this paper, we present three novel approaches for solving the MPE task in PASP that are based on: i) Knowledge compilation [Darwiche and Marquis, 2002], ii) Answer Set Programming (ASP) [Gelfond and Lifschitz, 1991; Brewka *et al.*, 2011], and iii) ASP with quantifiers (ASP(Q)) [Amendola *et al.*, 2019].

Knowledge compilation (KC) [Darwiche and Marquis, 2002] is a standard technique adopted in StarAI: the program is compiled into a form where the considered task can be solved more efficiently. After compilation, the initial problem boils down to solving a counting problem on the compiled form, which can often be expressed as Algebraic Model Counting (AMC) [Kimmig *et al.*, 2017]. The expressivity of some languages and the complexity of some tasks require stacking two layers of AMC (2AMC) [Kiesel *et al.*, 2022]. In this paper, we provide a 2AMC encoding of MPE as well as a practical implementation into the aspmc [Eiter *et al.*, 2021; Eiter *et al.*, 2024] solver. Then, we discuss an encoding in ASP which targets the computation of the upper MPE states only, that can be also adopted for the computation of the MPE state in probabilistic logic programs (i.e., stratified programs, where the lower and upper MPE states coincide). Lastly, building on the inherent complexity of the task, we also propose an encoding for computing the lower MPE state using ASP(Q) [Amendola *et al.*, 2019]. ASP(Q) extends traditional ASP by introducing the concept of quantifiers over answer sets that makes the modeling of problems in the entire Polynomial Hierarchy [Stockmeyer, 1976] as natural as modeling NP-hard problems in standard ASP. Our proposals have been assessed through an extensive evaluation considering multiple datasets targeting different probabilistic scenarios, and revealed to be the best performing ones in the case of non-stratified programs.

2 Background

Answer Set Programming. We assume the reader to be familiar with the basic concepts of Logic Programming. A *rule* r is an expression of the form $h_1; \ldots; h_n :- b_1, \ldots, b_m$ where $h_1; \ldots; h_n$, with $n \geq 0$, is a disjunction of atoms referred to as *head* and $b_1, \ldots, b_m$, with $m \geq 0$, is a conjunction of literals referred to as *body*. If $n = 0$ then r is a *strong constraint*; otherwise, if $m = 0$ then r is a *fact*. A *choice rule* is an expression of the form $\{a_1; \ldots; a_n\}$, with $n > 0$, where each a_i is a ground atom. Intuitively, a choice rule is a shorthand for a set of normal rules of the form "$a_i :- \, not \; na_i$", "$na_i :- \, not \; a_i$", where $\forall i \in \{1, \ldots, n\}$, na_i is a fresh atom not appearing elsewhere. Given a rule r, we denote by H_r the set of atoms appearing in the head and by B_r the set of literals appearing in the body. Given a set of literals L, we denote by L^+ (resp. L^-) the set of positive (resp. negative) literals in L. A *weak constraint* is an expression of the form $:\sim b_1, \ldots, b_m \; [w, T]$, where $b_1, \ldots, b_m$, with $m \geq 0$, is conjunction of literals, w is a term, and T is a (possibly empty) tuple of terms (in this case, T is omitted). An *answer set program* (ASP, and we use the same acronym to denote Answer Set Programming, the context will disambigaute the meaning) is a finite set of *safe* rules (i.e., variables appear at least in one positive literal of the body) and weak constraints. An ASP is *plain* if it contains no weak constraints. An ASP expression (i.e., literal, rules, etc.) is *ground* if it contains no variables.

Given a program P, the *Herbrand Universe*, denoted by $\mathcal{U}_P$, is the set of constants that appears in P. The *Herbrand Base*, denoted by $\mathcal{B}_P$, is the set of ground atoms that can be built from predicates appearing in P and constants in $\mathcal{U}_P$. Given a rule $r \in P$, $ground(r)$ denotes the set of ground instantiations of r obtained by substituting variables with constants in P. The grounding of P, denoted as $ground(P)$, is the union of the grounding of rules in P. The dependency graph of P, denoted by G_P, is a labeled directed graph whose nodes are atoms in $\mathcal{B}_P$ and there is a positive (resp. negative) edge from an atom b to an atom h if there exists $r \in ground(P)$ such that $b \in B_r^+$ (resp. $not\ b \in B_r^-$) and $h \in H_r$. A program P is said to be *stratified* if G_P does not contain loops involving negative edges, whereas P is *head-cycle-free*, if, for each $r \in P$ and each pair of atom $a, b \in H_r$, does not exist a positive loop in G_P involving both a and b. In what follows, ASP are assumed to be head-cycle-free.

The semantics of ASP is defined in terms of answer sets [Gelfond and Lifschitz, 1991]. More precisely, an answer set of a program P is an interpretation I (i.e., a subset of $\mathcal{B}_P$) that (i) satisfies all the rules in $ground(\Pi)$ (i.e., for each $r \in ground(P)$, $H_r \cap I \neq \emptyset$ whenever $B_r^+ \subseteq I$ and $\{a \mid not\ a \in B_r^-\} \cap I = \emptyset$); and (ii) I is a $\subset$-minimal model of its (Gelfond-Lifschitz) reduct [Gelfond and Lifschitz, 1991]. Let $AS(P)$ be the set of answer sets of P, then P is *coherent* if $AS(P) \neq \emptyset$; otherwise it is *incoherent*. The cost of an answer set M is defined as $\mathcal{C}(P, M) = \sum_{(w,T) \in ws(P,M)} w$, where $ws(P, M) = \{(w, T) \mid :\sim b_1, \dots, b_n, [w, T] \in ground(P)$, w is a numeric constant, and $M \models b_1, \dots, b_n\}$. An answer set $M \in AS(P)$ is *optimal* if there does not exist $M' \in AS(P)$ having a strictly lower cost (i.e., such that M is *dominated* by M'). For further details on the ASP semantics, we refer the reader to the dedicated literature [Gelfond and Lifschitz, 1991; Buccafurri *et al.*, 2000; Calimeri *et al.*, 2020].

Example 1. *Let us consider the following program:*

```
a :- not na. na :- not a. :~ a. [1]
```

Here we have two answer sets $M_1 = \{a\}$ and $M_2 = \{na\}$. Since a is true in M_1, its cost is 1. *Conversely, since a is false in M_2, its cost is* 0. *Thus, only M_2 is optimal.*

Answer Set Programming with Quantifiers. ASP with Quantifiers (ASP(Q)) extends ASP by introducing quantifiers over answer sets [Amendola *et al.*, 2019; Mazzotta *et al.*, 2024]. Namely, an ASP(Q) Π is an expression of the form:

$$\Box_1 P_1 \cdots \Box_n P_n : C : C^\omega \tag{1}$$

where for each $i \in \{1, \dots, n\}$, $\Box_i \in \{\exists^{st}, \forall^{st}\}$ and P_i is a plain ASP, C is a plain stratified program with strong constraints, and C^ω is a set of weak constraints. An ASP(Q) is said to be *existential* if $\Box_1 = \exists^{st}$; otherwise, it is *universal*. The *coherence* of an ASP(Q) Π is defined as follows:

- $\exists^{st} P : C : C^\omega$ is coherent, if there exists $M \in AS(P)$ such that $C \cup \mathit{fix}_P(M)$ admits an answer set;
- $\forall^{st} P : C : C^\omega$ is coherent, if for every $M \in AS(P)$, $C \cup \mathit{fix}_P(M)$ admits an answer set;
- $\exists^{st} P\ \Pi'$ is coherent, if there exists $M \in AS(P)$ such that $\Pi'_{P,M}$ is coherent;
- $\forall^{st} P\ \Pi'$ is coherent, if for every $M \in AS(P)$, $\Pi'_{P,M}$ is coherent.

where Π' is an ASP(Q) of the form (1), $\Pi_{P,M}$ denotes the ASP(Q) obtained from Π by replacing P_1 with $P_1 \cup \mathit{fix}_P(M)$ (i.e., $\Pi_{P,M} = \Box_1(P_1 \cup \mathit{fix}_P(M))\, \Box_2 P_2 \cdots \Box_n P_n : C : C^\omega$), and $\mathit{fix}_P(M)$ denotes the set of facts and strong constraints $\{a \mid a \in M\} \cup \{\leftarrow a \mid a \in \mathcal{B}_P \setminus M\}$. For an existential ASP(Q) Π, $M \in AS(P_1)$ is a *quantified answer set* of Π if $(\Box_2 P_2 \cdots \Box_n P_n : C : C^\omega)_{P_1,M}$ is coherent. We denote by $QAS(\Pi)$ the set of quantified answer sets of Π.

For an ASP(Q) of the form (1), the set of weak constraints C^ω, referred to as *global weak constraints*, are used for specifying preferences criteria over quantified answer sets. More formally, let Π be an existential ASP(Q) of the form (1) and $M_1, M_2 \in QAS(\Pi)$. We say that M_1 is *dominated* by M_2 if $\mathcal{C}(P_1^*, M_2) < \mathcal{C}(P_1^*, M_1)$ where $P_1^* = P_1 \cup C^\omega$. A quantified answer set $M \in QAS(\Pi)$ is *optimal* if does not exist $M' \in QAS(\Pi)$ such that M is dominated by M'.

Example 2. *Let us consider an ASP(Q) $\Pi = \exists P_1 \forall P_2 : C : C^\omega$, where P_1 is the ASP from Example 1, P_2 is the program:*

```
b :- not nb. nb :- not b.
```

$C = \{ :- a, nb\}$, and $C^\omega = :\sim a$. From Example 1, P_1 admits M_1 and M_2 as answer sets. If we fix M_1 in P_2 we obtain two answer sets, $M_3 = \{a, b\}$ and $M_4 = \{a, nb\}$. Since M_4 violates the constraint in C then M_1 is not a quantified answer set. Instead, by fixing M_2 in P_2 we obtain two answer sets, $M_5 = \{na, b\}$ and $M_6 = \{na, nb\}$. Since both M_5 and M_6 satisfy the constraint in C, then M_2 is the only quantified answer set for Π *and so it is also optimal.*

Probabilistic Answer Set Programming. One possible formalism to encode uncertain data with a logic-based language is Probabilistic Answer Set Programming under the credal semantics [Cozman and Mauá, 2020; Mauá and Cozman, 2020], PASP, for short. A PASP is an ASP extended with probabilistic facts that follows the syntax [De Raedt *et al.*, 2007] $p :: a$ with $p \in [0, 1]$ and a an atom. We assume that probabilistic facts cannot appear as the head of rules. For a PASP $\mathcal{P}$, we denote with $\mathbb{F}(\mathcal{P})$ the set of atoms $\{a \mid p :: a \in \mathcal{P}\}$ and with $\mathbb{P}(\mathcal{P})$ the underlying ASP. A *selection* σ is a set of atoms associated with probabilistic facts, i.e., $\sigma \subseteq \mathbb{F}(\mathcal{P})$. We denote by Σ the set of all selections, i.e., $\Sigma = 2^{\mathbb{F}(\mathcal{P})}$. The probability of a selection σ is computed as $P(\sigma) = \prod_{a_i \in \sigma} p_i \prod_{a_i \notin \sigma}(1 - p_i)$. A selection σ identifies a *world* ψ_σ composed of the rules of $\mathcal{P}$ and a fact a_i for each $a_i \in \sigma$. The probability of a world ψ is equal to the probability of the corresponding selection, i.e., $P(\psi_\sigma) = P(\sigma)$. To simplify the notation, when it is clear from the context, we will use $P(\psi)$ and drop the subscript. The log-probability of a selection can be computed by considering summations as $log(P(\sigma)) = \sum_{a_i \in \sigma} log(p_i) + \sum_{a_i \notin \sigma} log(1 - p_i)$. Then, the probability of a world ψ, $P(\psi)$, can be computed from $log(P(\psi))$ as $e^{log(P(\psi))}$. A PASP $\mathcal{P}$ with n probabilistic facts in $ground(\mathcal{P})$ has 2^n worlds, and the credal semantics requires that each world has at least one answer set. A *query* is a conjunction of ground literals. The probability of a query q belongs to a range $[\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)]$, where the two bounds are called *lower* and *upper* probability, respectively, with $\underline{\mathrm{P}}(q) = \sum_{\psi_i \mid \forall M \in AS(\psi_i),\ M \models q} P(\psi_i)$ and $\overline{\mathrm{P}}(q) = \sum_{\psi_i \mid \exists M \in AS(\psi_i),\ M \models q} P(\psi_i)$. Note that $|AS(\psi_i)| > 0$ as required by the credal semantics. That is, the worlds where the query is true in *every* answer set (i.e., where the query is cautiously entailed) contribute to both the lower and the upper probability, while the worlds where the query is true only in *some* answer sets (i.e., the query is bravely entailed) contribute only to the upper probability. To clarify, consider the following example.

Example 3. *The following program is one of the possible encodings for the graph 3-coloring task.*

```
r(X) :- n(X), not g(X), not b(X).
g(X) :- n(X), not r(X), not b(X).
b(X) :- n(X), not r(X), not g(X).
edg(X,Y) :- e(X,Y).
edg(X,Y) :- e(Y,X).
:- edg(X,Y), r(X), r(Y).
:- edg(X,Y), g(X), g(Y).
:- edg(X,Y), b(X), b(Y).
```

Consider a graph with four nodes, where two of them have a fixed color:

```
n(1). n(2). n(3). n(4). r(1). g(4).
```

Suppose that we are uncertain about the presence of edges between nodes. We describe this with the probabilistic facts:

```
0.6::e(1,2). 0.1::e(1,3).
0.2::e(2,4). 0.7::e(3,4).
```

Also suppose that we want to compute the probability that blue is assigned to at least one of the two uncolored nodes. This can be encoded with the rules $blue :- b(2)$ and $blue :- b(3)$ and by asking for the probability of blue. This program has $2^4 = 16$ worlds. For example, in the world where all edges except $e(1,2)$ are present, we have two answer sets representing two possible color assignments for node 2, either red or blue, while the color of node 3 is fixed to blue in both. So, this world contributes to both the lower and upper probability. Overall, the probability of blue is $[0.1816, 1]$.

MPE in PASP. Let us introduce the MPE problem in PASP.

Definition 1. *Given a PASP $\mathcal{P}$ and a conjunction of ground literals e called evidence, the* cautious MPE *problem consists in finding the selection σ with the highest associated probability where e is entailed in every answer set of the corresponding world ψ_σ, i.e.:*

$$\underline{\mathrm{MPE}}(e) = \underset{\sigma \in \Sigma \mid \forall M \in AS(\psi_\sigma), M \models e}{\arg\max} P(\sigma),$$

while the brave MPE *problem consists in finding the selection σ with the highest associated probability where the e is entailed in at least one answer set of the corresponding world ψ_σ, i.e.:*

$$\overline{\mathrm{MPE}}(e) = \underset{\sigma \in \Sigma \mid \exists M \in AS(\psi_\sigma), M \models e}{\arg\max} P(\sigma).$$

The results of the two aforementioned problems are respectively called *lower* and *upper MPE states*. We use $P(\underline{\mathrm{MPE}}(e))$ and $P(\overline{\mathrm{MPE}}(e))$ to denote the probability of the selection (world) representing the lower and upper MPE state, respectively, and explicitly mark the atoms not present in such states by prepending them with *not*. In what follows, we assume w.l.o.g. that e is a single atom. Indeed, it is possible to add a rule with the conjunction of literals in e in the body and as head a new atom e' that does not appear elsewhere in the program and considering as evidence e' (as in Example 3).

Example 4. *Consider again Example 3. If we consider as evidence blue as defined in Example 3, we obtain* $\underline{\mathrm{MPE}}(blue) = \{e(1,2),\ not\ e(1,3),\ e(2,4),\ e(3,4)\}$ *with* $P(\underline{\mathrm{MPE}}(blue)) = 0.0756$ *and* $\overline{\mathrm{MPE}}(blue) = \{e(1,2), not\ e(1,3), not\ e(2,4), e(3,4)\}$ *with* $P(\overline{\mathrm{MPE}}(blue)) = 0.3024$. *Note that, as in this example, the lower and upper MPE states may not coincide.*

Theoretical results from [Mauá and Cozman, 2020] proved that the MPE problem is hard (i.e., for propositional PASP, it lies between NP and Σ_3^p, depending on the considered ASP fragment) and early approaches to solve this task [Azzolini *et al.*, 2023] were based on projected answer set enumeration [Gebser *et al.*, 2009]. We now provide novel alternative ways to handle the MPE computation.

3 Algebraic Representation of MPE

Second-level algebraic model counting (2AMC) [Kiesel *et al.*, 2022] is a generic framework that extends algebraic model counting [Kimmig *et al.*, 2017].

Definition 2 (2AMC). *Given a propositional theory Π whose variables are partitioned into (V_i, V_o), two commutative semirings [Gondran and Minoux, 2008] $\mathcal{R}_i = (R^i, \oplus^i, \otimes^i, n_\oplus^i, n_\otimes^i)$ and $\mathcal{R}_o = (R^o, \oplus^o, \otimes^o, n_\oplus^o, n_\otimes^o)$, two weight functions, w_i and w_o, and a transformation function f_{io} mapping the values of R^i to R^o, let T be $T = (\Pi, V_i, V_o, \mathcal{R}_i, \mathcal{R}_o, w_i, w_o, f_{io})$. 2AMC on T is defined as:*

$$2AMC(T) = \bigoplus\nolimits^o_{I_o \in \mu(V_o)} \bigotimes\nolimits^o_{a \in I_o} w_o(a) \otimes^o f_{io}\Big(\bigoplus\nolimits^i_{I_i \in \varphi(\Pi \mid I_o)} \bigotimes\nolimits^i_{b \in I_i} w_i(b)\Big) \quad (2)$$

where $\mu(V_o)$ is the set of possible assignments to V_o and $\varphi(\Pi \mid I_o)$ the set of assignments I_i of V_i such that (I_i, I_o) satisfies Π.

Each of the two layers in 2AMC is a tuple $A_x = (V_x, \mathcal{R}_X, w_x)$. So, there is an inner layer $A_i = (V_i, \mathcal{R}_i, w_i)$ whose values are mapped via a transformation function f_{io} to values of the outer layer $A_o = (V_o, \mathcal{R}_o, w_o)$. MPE inference in PASP can be cast as a 2AMC, as follows.

Definition 3 (MPE as 2AMC)**.** *Consider a PASP $\mathcal{P}$ with Herbrand base $\mathcal{B}_\mathcal{P}$ and evidence e. $\mathcal{P}$ is transformed into a propositional theory Π with a one-to-one correspondence between the answer sets and the models of Π. This is achieved with the steps of grounding, simplification, cycle breaking, and Clark's completion, as described in [Eiter* et al.*, 2024]. For the innermost layer A_i we have [Azzolini and Riguzzi, 2023a]: as semiring, $\mathcal{R}^i = (\mathbb{N}^2, +, \cdot, (0,0), (1,1))$, where $+$ and $\cdot$ are applied component-wise, as variables $\mathcal{B}_\mathcal{P} \setminus \mathbb{F}(\mathcal{P})$, as weight function:*

$$w_i(a) = \begin{cases} (0,1) & \text{if } a = not\ e, \\ (1,1) & \text{otherwise.} \end{cases}$$

The two components store, respectively, the count of the answer sets where the query is true and the count of all the answer sets. As transformation function, we consider $f_{io}(n_1, n_2) = (v_{lp}, v_{up}, \{\}, \{\})$ where $v_{lp} = 1$ if $n_1 = n_2$, 0 otherwise, and $v_{up} = 1$ if $n_1 > 0$, 0 otherwise.

For the outer layer A_o we have the semiring $\mathcal{R}_o = (\mathbb{R}^2 \times \Sigma^2, max^4, times^4, (0,0,\mathbb{F}(\mathcal{P}),\mathbb{F}(\mathcal{P})), (1,1,\{\},\{\}))$, as variables $\mathbb{F}(\mathcal{P})$, and as weight function

$$w_o(a) = \begin{cases} (p, p, \{a\}, \{a\}) & \text{if } a = f,\ p :: f, \\ (1-p, 1-p, \{a\}, \{a\}) & \text{if } a = not\ f,\ p :: f, \\ (1, 1, \{a\}, \{a\}) & \text{otherwise.} \end{cases}$$

where $max^4((v_0, v_1, S_0, S_1), (v_a, v_b, S_a, S_b)) = (v_x, v_y, S_x, S_y)$ with $v_x = v_0$ and $S_x = S_0$ if $v_0 > v_a$, else $v_x = v_a$ and $S_x = S_a$; $v_y = v_1$ and $S_y = S_1$ if $v_1 > v_b$, else $v_y = v_b$ and $S_y = S_b$. $times^4((v_0, v_1, S_0, S_1), (v_a, v_b, S_a, S_b)) = (v_0 \cdot v_a, v_1 \cdot v_b, S_0 \cup S_a, S_1 \cup S_b)$. If $v_0 = v_a$, then $S_x = min_>(S_0, S_a)$; if $v_1 = v_b$, then $S_y = min_>(S_1, S_b)$, where, in both cases, $>$ is a fixed total order on Σ. The first two components store, respectively, the value associated with the lower and upper MPE states while the last two store the probabilistic facts present in such states.

In other words, in the inner layer, we have the probabilistic facts fixed, and we need to compute whether the current world contributes to the lower or upper MPE state (i.e., if the evidence is cautiously or bravely entailed) while in the outer layer we range over the possible worlds.

4 Logic Based Encodings for MPE

In this section, we propose logic-based encodings to solve the MPE problem in PASP. In particular, we propose a natural ASP(Q) encoding to solve the cautious MPE problem and an ASP one to solve the brave MPE problem. Both rely on the following encoding of probabilistic facts as ASP rules.

Definition 4. *Let $\mathcal{P}$ be a PASP. $worlds(\mathcal{P})$ denotes the program obtained by encoding every $p :: a \in \mathcal{P}$ as follows:*

$$\{a\}. \quad prob(w, id_a) :- a. \quad prob(\overline{w}, id_a) :- not\ a.$$

where id_a is a unique identifier for the probabilistic fact $p :: a$, $w = log(p)$, and $\overline{w} = log(1-p)$.

In this definition, choice rules encode possible selections σ while normal rules derive an atom storing either $log(p)$ if a is chosen in σ; otherwise $log(1-p)$, for each $p :: a \in \mathcal{P}$.

4.1 Cautious MPE Problem in ASP(Q)

Given a PASP $\mathcal{P}$ and evidence e, we encode the $\underline{\text{MPE}}(e)$ task by means of an existential ASP(Q) such that quantified answer sets are in one-to-one correspondence with selections that cautiously entail the evidence e and the optimal quantified answer sets correspond to lower MPE states.

Definition 5. *Let $\mathcal{P}$ be a PASP and e be the evidence, then $aspq(\mathcal{P}, e) = \exists^{st} P_1\ \forall^{st} P_2 : C : C^\omega$, where $P_1 = worlds(\mathcal{P})$, $P_2 = \mathbb{P}(\mathcal{P})$, $C = \{\leftarrow not\ e\}$, and $C^\omega = \{:\sim prob(W, A)\ [-W, A]\}$.*

By exploiting such an encoding, each optimal quantified answer set M of $aspq(\mathcal{P}, e)$ corresponds to a lower MPE state in which all the atoms in $\mathbb{F}(\mathcal{P}) \cap M$ are chosen to be true and the remaining ones in $\mathbb{F}(\mathcal{P})$ are false. Thus, all the lower MPE states can be obtained by enumerating optimal quantified answer sets. The associated probability can be obtained by considering the probabilities of the facts selected in the quantified answer set.

Theorem 1. *Let $\mathcal{P}$ be a PASP, e be an evidence, and $\Pi = aspq(\mathcal{P}, e)$, then M is an optimal quantified answer set of Π iff $\sigma = M \cap \mathbb{F}(\mathcal{P})$ is a lower MPE state.*

Proof Sketch. ($\Rightarrow$) By construction, each answer set M_1 of P_1 corresponds to a selection $\sigma = M_1 \cap \mathbb{F}(\mathcal{P})$. Then, by fixing M_1 in the program P_2, answer sets of the program $P_2 \cup fix_{P_1}(M_1)$ are in one-to-one correspondence with answer sets of the world ψ_σ identified by σ. According to the coherence of ASP(Q), if for every $M_2 \in AS(P_2 \cup fix_{P_1}(M_1))$ the constraint $:-\ not\ e$ is satisfied, then M_1 is a quantified answer set. This means that for every answer set of ψ_σ the evidence is entailed and so ψ_σ cautiously entails e. Moreover, the cost of M_1 is obtained by summing up $-W$ for each atom of the form $prob(W, Id)$ that is true w.r.t. M_1. By construction, for each $p :: a \in \mathcal{P}$, either $prob(w, id_a)$ or $prob(\overline{w}, id_a)$ is true w.r.t. M_1 and so the cost of M_1 is equal to $-log(P(\sigma))$. Since an optimal quantified answer set minimizes such a cost, we are indeed maximizing $log(P(\sigma))$, and, if M_1 is optimal, then σ is a lower MPE state.

($\Leftarrow$) Here the proof follows by noting that each lower MPE state σ could be mapped to an optimal quantified answer set $M = \sigma \cup \{prob(log(p), id_a) \mid p :: a \in \mathcal{P}, a \in \sigma\} \cup \{prob(log(1-p), id_a) \mid p :: a \in \mathcal{P}, a \notin \sigma\}$.

Example 5. *Let us consider the PASP problem of Example 3. We can encode $\underline{\text{MPE}}(blue)$ with the following ASP(Q):*

```
%@exists
{e(1,2);e(1,3);e(2,4);e(3,4)}.
prob(log(0.6),e12):- e(1,2).
prob(log(0.4),e12):- not e(1,2).
prob(log(0.1),e13):- e(1,3).
prob(log(0.9),e13):- not e(1,3).
prob(log(0.2),e24):- e(2,4).
prob(log(0.8),e24):- not e(2,4).
prob(log(0.7),e34):- e(3,4).
prob(log(0.3),e34):- not e(3,4).
%@forall
r(X) :- n(X), not g(X), not b(X).
g(X) :- n(X), not r(X), not b(X).
b(X) :- n(X), not r(X), not g(X).
```

```
edg(X,Y) :- e(X,Y). edg(X,Y) :- e(Y,X).
:- edg(X,Y), r(X), r(Y).
:- edg(X,Y), g(X), g(Y).
:- edg(X,Y), b(X), b(Y).
blue :- b(2). blue :- b(3).
n(1..4). r(1). g(4).
%@constraint
:- not blue.
%@global
:~ prob(W,A). [-W,A]
```

Answer sets of P_1 represent selections with their associated probability. For example, consider M_1=$\{e(1,2)$, $e(1,3)$, $e(3,4)$, $prob(log(0.6), e12)$, $prob(log(0.1), e13)$, $prob(log(0.8), e24)$, $prob(log(0.7), e34)\}$ $\in$ $AS(P_1)$. By fixing M_1 into P_2, we obtain two answer sets of $P_2 \cup fix_{P_1}(M_1)$, where in the first both nodes 2 and 3 are blue, while in the second nodes 2 and 3 are green and blue, respectively, which are indeed the answer set of the world ψ_σ with $\sigma = \{e(1,2), e(1,3), e(3,4)\}$. Since in both cases blue is true, the constraint in C is satisfied, M_1 is a quantified answer set, and σ cautiously entails blue. The cost of M_1 is $-log(P(\sigma)) \approx 3.3932$. Similarly, $M_1' = \{e(1,2)$, $e(2,4)$, $e(3,4)$, $prob(log(0.6), e12)$, $prob(log(0.9), e13)$, $prob(log(0.2), e24)$, $prob(log(0.7), e34)\}$ $\in$ $AS(P_1)$ is a quantified answer set. M_1' is associated with the selection $\sigma' = \{e(1,2), e(2,4), e(3,4)\}$ so the cost of M_1' is $-log(P(\sigma')) \approx 2.5829$. M_1 is dominated by M_1', and so not optimal. No better quantified answer set exists, so M_1' is optimal and σ' is a lower MPE state with probability 0.0756.

4.2 Brave MPE Problem in ASP

The brave MPE problem can be encoded with an ASP whose optimal answer sets correspond with upper MPE states.

Definition 6. *Let $\mathcal{P}$ be a PASP and e be an evidence, then $asp(\mathcal{P}, e)$ is the program obtained from $worlds(\mathcal{P})$ by adding $\mathbb{P}(\mathcal{P})$, the constraint $:-$ not e, and the weak constraint $:\sim prob(W, Id)$ $[-W, Id]$.*

Intuitively, rules in $worlds(\mathcal{P})$ model the search for a possible selection and rules in $\mathbb{P}(\mathcal{P})$ model possible worlds. Finally, the constraint $:-$ *not* e forces the entailment of the evidence, whereas the weak constraint allows the maximization of the sum of the log-probability associated with probabilistic facts. Thus, to obtain all the upper MPE states it is sufficient to enumerate all the optimal answer sets of $asp(\mathcal{P}, e)$.

Theorem 2. *Let $\mathcal{P}$ be a PASP, e be an evidence, and $P = asp(\mathcal{P}, e)$. Then, M is an optimal answer set for P if and only if $\sigma = M \cap \mathbb{F}(\mathcal{P})$ is an upper MPE state.*

Proof Sketch. ($\Rightarrow$) By construction, there is a one-to-one correspondence between the answer sets of worlds ψ_σ that bravely entails the evidence e and the answer sets of $asp(\mathcal{P}, e)$. Thus, for each $M \in AS(asp(\mathcal{P}, e))$, there exists a selection $\sigma = M \cap \mathbb{F}(\mathcal{P})$ such that, by removing atoms over predicate $prob/2$ from M, we obtain $M' \in AS(\psi_\sigma)$ such that $M' \models e$. Moreover, each answer set M of $asp(\mathcal{P}, e)$ has an associated cost that, as for the ASP(Q) encoding, is obtained by summing up $-W$ for each atom of the form $prob(W, Id)$ that is true w.r.t. M. Thus, as for the previous encoding, we are indeed maximizing $log(P(\sigma))$ and so an optimal answer set corresponds to an upper MPE state.

($\Leftarrow$) Here the proof follows by noting that an upper MPE state σ could be mapped to an optimal answer set $M = M' \cup \{prob(log(p), id_a) \mid p :: a \in \mathcal{P}, a \in \sigma\} \cup \{prob(log(1 - p), id_a) \mid p :: a \in \mathcal{P}, a \notin \sigma\}$, where $M' \in AS(\psi_\sigma)$.

Example 6. *Let $\mathcal{P}$ be the PASP of Example 3. $\overline{\text{MPE}}(blue)$ can be encoded as $asp(\mathcal{P}, blue)$. The resulting ASP is the ASP(Q) of Example 5 where all subprograms are merged into a unique ASP. For the sake of compactness, we avoid reporting it explicitly. Here, $M_1 = \{n(1)$, $n(2)$, $n(3)$, $n(4)$, $g(4)$, $r(1)$, $e(1,2)$, $e(3,4)$, $prob(log(0.6), e12)$, $prob(log(0.9), e13)$, $prob(log(0.8), e24)$, $prob(log(0.7), e34)$, $b(2)$, $b(3)$, $blue$, $edg(1,2)$, $edg(3,4)$, $edg(2,1)$, $edg(4,3)\}$ is an optimal answer set of $asp(\mathcal{P}, blue)$. M_1 can be mapped to the selection $\sigma = \{e(1,2), e(3,4)\}$. By removing atoms over the predicate $prob/2$ from M_1, we can obtain $M_1' \in AS(\psi_\sigma)$ such that $M_1' \models blue$. The cost of $M_1 = -log(P(\sigma)) \approx 1.1960$. No answer sets better than M_1 exists, so σ is an upper MPE state with probability 0.3024.*

5 Experimental Evaluation

The experiments were executed on a machine running at 3.7 GHz with 32 GB of RAM and a time limit, for each instance, of 3600 seconds (1 hour). We used the bash command *time* to extract the execution time and collected the *real* values. Source code and datasets are available at https://t.ly/kqulc.

5.1 Datasets Description

We consider two types of datasets: stratified and non-stratified. For both types, we randomly generated instances of increasing size (a fixed seed ensures reproducibility).

Stratified Dataset. Stratified dataset refers to PASP where the underlying ASP is a stratified program. Thus, in this category we considered the well-known reachability problem, with graphs following three different structures: Barabási Albert, Erdős-Rényi, and complete. For each type, we generated directed graphs with an increasing number of nodes (considered as the size of the instance), starting from 2 and up to 100. We considered two variations, having the same edges but different probabilities: fixed to 0.4 and random (uniform) between 0.001 and 0.999. We obtained 588 programs in total.

Non-Stratified Datasets. Here, we considered six different benchmarks. For all, the evidence is the atom qr. For each of the first four benchmarks, an instance of size i has a total number of i probabilistic facts having the signature $aj/0$ with $j \in \{0, \ldots, i-1\}$. An instance of size i of the dataset $p1$ has $i/2$ rules of the form $qr_j; nqr_j :- \diamond a_j$ with j even where $\diamond$ can be either *not* or omitted, $i/2$ rules of the form $qr_{k-1} :- \diamond a_k$ with k odd and $\diamond$ as before, and a rule $qr :- \bigwedge qr_i$ with i even. An instance of size i of the dataset $p2$ has $i/2$ rules of the form $qr_j; nqr_j :- \circ, \diamond a_j$ with j even where $\circ$ can be qr_{k-1} or nqr_{k-1} and $\diamond$ can be *not* or omitted (except when $j = 0$, where *not* is omitted), $i/2$ rules of the form $qr_k :- \circ, \diamond a_k$ with k odd and the rest as before, and i rules $qr :- qr_i$. An instance of size i of the dataset $p3$ has i

rules of the form $\bigvee_{j\in\{0,i-1\}} qr_j :- \diamond a_i$ and a rule $qr :- qr_j$ for each $j \in \{0, i-1\}$. An instance of size i of the dataset $p4$ has one rule with head $qr; nqr$ and body composed of a conjunction of random probabilistic facts where each one can be also considered as negated, and a rule with head qr and body as before. Note that here some of the probabilistic facts may not be considered in the two rules. For these benchmarks we randomly generated instances with sizes from 5 up to 100 by sampling values for $\circ$ and $\diamond$ with uniform distributions. In addition, we considered both the well known graph coloring (*col*) problem where the edges are considered as probabilistic facts and the evidence is the color of a given node, and the friend-smokers (*smk*) benchmark [Totis *et al.*, 2023], representing a network of people where the smoking behavior is influenced by friendship and existing pathologies. Here, the evidence encodes whether at least one individual smokes. An instance of size i of *col* has i probabilistic facts and encodes a graph with i nodes. We generated instances starting with size 7 and up to 100 by adding random edges between nodes. An instance of size i of smokers involves i people and has $5i - 1$ probabilistic facts in total. We generated instances starting from size 2 and up to 30, also by adding pathologies and connections between people. In all instances, we ensured that every world has at least one answer set (as required by the credal semantics) and the lower MPE state always exists. Also here we consider two variations, with probabilities fixed to 0.4 and random. We have 1014 programs in total.

5.2 Compared Algorithms

In these experiments, we compare the proposed approaches with the ones available in the literature: PASTA [Azzolini *et al.*, 2022; Azzolini and Riguzzi, 2023b], cplint [Bellodi *et al.*, 2020], ProbLog [Shterionov *et al.*, 2015], and plingo [Hahn *et al.*, 2024]. plingo is a recently introduced system to perform inference in a (subset of) ProbLog programs using the ASP solver clingo [Gebser *et al.*, 2019] as a backend. Essentially, plingo rewrites weighted rules [Lee and Yang, 2017] into ASP and then uses clingo to compute answer sets. To model the MPE problem in plingo we need to specify probabilistic facts with external atoms in this way: for a probabilistic fact $p :: a$, we add $a :- \&problog(p)$. Furthermore, the evidence e is encoded as $\&evidence(e)$. We implemented the 2AMC method of Section 3 into the aspmc [Eiter *et al.*, 2021; Eiter *et al.*, 2024] solver, that we indicate with aspmc in the following. For ASP, we used the solver clingo with the encoding of Section 4.2. In ASP and ASP(Q) implementations, we discretize log-probabilities as $\lfloor k \cdot log(p) \rfloor$ and use $k = 10^3$. Moreover, for both ASP and plingo we used the clingo option `--opt-strategy=usc` [Andres *et al.*, 2012] which revealed to be an effective strategy for both solvers.

5.3 Results

The goal of our experiments was to empirically answer the following questions: **Q1**) is knowledge compilation effective in solving MPE in PASP? **Q2**) is ASP a good candidate for MPE when considering stratified programs? **Q3**) does the probability associated with facts influence the runtime? **Q4**) is ASP(Q) a good approach for solving MPE in PASP? **Q5**) does the structure of the program influence the runtime?

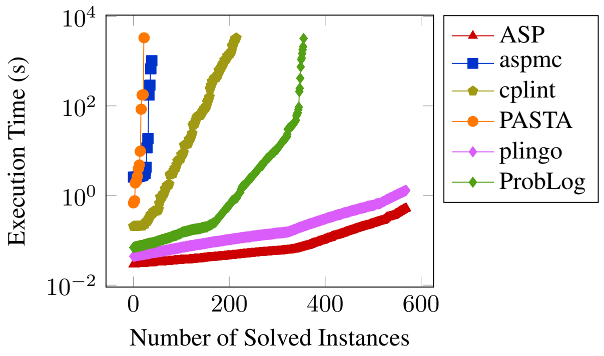

Figure 1: Cactus plot for stratified datasets (y log scale).

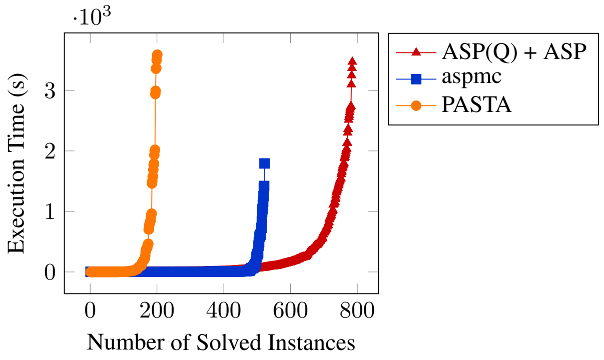

Figure 2: Cactus plot for non-stratified datasets.

Results on stratified datasets. In these datasets, every instance has a unique answer set per world, so lower and upper probability coincide and also lower and upper MPE states (that we simply call MPE states). In this setting, both logic-based approaches can be used but, based on their own complexity, we consider only the ASP-based one. Concerning cplint and ProbLog, instead, it is important to point out that if the program contains probabilistic facts that are not relevant to the computation of the probability, these facts are ignored. Thus, the states these two systems find are not the MPE states, but they can be straightforwardly extended to be so. If $S^{'}$ is the state computed with either one of the two tools, the MPE state is given by $S = S^{'} \cup \bigcup_{a_i \in F \cap S^{'}} p(a_i)$ where for a probabilistic fact $p_i :: a_i$, $p(a_i)$ returns $\{a_i\}$ if $p_i > 0.5$, else it returns $\{\}$, and the probability of such a state can be computed accordingly.

Figure 1 reports the cactus plot for the experiments on stratified datasets. We recall that instances in a cactus plot are sorted by execution time and a point (x, y) indicates that a given system solved the x-th instance in y seconds. PASTA is the slowest algorithm, being it based on (projected) enumeration. aspmc closely follows it, despite being based on knowledge compilation. This is probably due to the grounding phase, which also considers parts of the program not relevant to the computation of the probability of the evidence. This has a high impact, in particular on Barabási Albert graphs, where often only a few edges are relevant for the computation. ProbLog can solve approximately twice the instances of

	aspmc		PASTA		ASP(Q) + ASP	
Dataset	sz.	t.	sz.	t.	sz.	t.
$p1^r$	33	4614	23	3675	23*	878
$p1^f$	33	4351	24	6731	22	70
$p2^r$	100	530	23	3846	84	21006
$p2^f$	100	530	23	3886	100	167
$p3^r$	20	2428	23	3401	89*	28044
$p3^f$	20	2329	23	3432	100	3072
$p4^r$	100	408	24	6422	100	52354
$p4^f$	100	407	24	6445	100	215
col^r	24	2565	23	5475	37	12713
col^f	24	2813	23	5742	100	4592
smk^r	7	884	5	3697	18	3142
smk^f	7	879	5	3683	69	52452

Table 1: Largest solvable instance (sz.) and execution time in seconds on all instances (t.) on non-stratified datasets. The $*$ in 23 of $p1^r$ in ASP(Q) + ASP states that the solver solves up to size 21, cannot solve size 22, and solves size 23. Similarly for size 89 of $p3^r$.

cplint. ASP and plingo are the best performing approaches with comparable execution times. For both, even the largest instance is solved in approximately 1 second. This provides a partial positive answer to **Q1** and positive answer to **Q2**. KC in ProbLog and cplint is more effective than KC in aspmc, since it considers only the relevant part of the program.

Results on non-stratified datasets. In non-stratified datasets each world may have one or more answer sets, thus, we compare only aspmc, PASTA, and ASP(Q) + ASP on the computation of both lower and upper MPE states. Our approaches based on ASP(Q) and 2AMC are, to the best of our knowledge, the only ones, in addition to PASTA, that can compute the lower MPE state in PASP. Note that, aspmc and PASTA compute both upper and lower MPE states in one single call, while ASP(Q) and ASP computes only the lower and upper MPE states, respectively. Thus, for a fair comparison, we sum the execution times of the two for each instance. However, it is important to point out that the execution time of ASP is negligible (less than 1 second) and solves all of them, so the discussion below mainly considers ASP(Q).

Table 1 shows the largest solvable instance and the cumulative execution time on each dataset. The superscripts r and f denote the instances with random and fixed associated probability, respectively. For PASTA the total execution time and the number of instances solved seems not to be affected by the choice of probability, except for $p1^f$: here, the instance of size 24 is solvable within the time limits while the instance of same size with random probabilities cannot be solved. The independence between the choice of the probability and of the execution time is expected in PASTA, since it adopts enumeration and the number of generated answer sets does not depend on the probabilities associated with probabilistic facts. PASTA always halts for the time limit. For aspmc, the execution always halts due to an out-of-memory error. The number of solvable instances in ASP(Q), differently from the other solvers, is highly dependent from the probabilities associated to probabilistic facts: on four datasets with fixed probabilities, it can solve all the instances, while this does not happen when random probabilities are considered. This is due to the upper-bound improving algorithm employed by the ASP(Q) system which starts from a quantified answer set with a given cost and searches for a better one until the program becomes unsatisfiable. In this scenario, proving unsatisfiability can be hard, especially when many worlds have costs that are better than the current upper-bound but none of them cautiously entail the query (i.e., they do not contribute to lower MPE state). ASP(Q) always halts due to timeout. For some datasets, ASP(Q) has dramatically better performance: in $p3^f$, aspmc stops at size 20, PASTA gets to size 23, while ASP(Q) can solve all. This is reflected in the total number of solved instances, as shown in the cactus plot of Figure 2. This answers positively **Q4** in general and **Q3** for ASP(Q) (but not for PASTA and aspmc). Overall, KC is also beneficial here (**Q1**), but the clever encoding of ASP(Q) that prunes dominated solutions during the search scales better.

6 Related Work

Related work falls into two categories: alternative semantics for ASP extended with constructs representing uncertainty and alternative MPE encodings. Other semantics, still targeting ASP, are: P-log [Baral *et al.*, 2009], LP$^{\text{MLN}}$ [Lee and Wang, 2016; Lee and Yang, 2017], smProbLog [Totis *et al.*, 2023], and the L-credal semantics [Rocha and Gagliardi Cozman, 2022; Mauá *et al.*, 2024]. These differ on how the uncertainty is distributed among the different answer sets of the program. dPASP [Geh *et al.*, 2024] is a tool to perform inference in PASP, but it does not support MPE. The authors of [Lee and Yang, 2017] were the first to propose the use of weak constraints to compute the most probable stable model, albeit in the LP$^{\text{MLN}}$ semantics. Alternative encodings can be used to model MPE in PASP, among these, disjunctive ASP and advanced techniques such *saturation* [Eiter and Gottlob, 1995; Dantsin *et al.*, 2001], but they can be considered less intuitive than ASP(Q). Moreover, the stable-unstable semantics [Bogaerts *et al.*, 2016], and quantified answer set semantics [Fandinno *et al.*, 2021], can be also used to model problems of such complexity but they do not support explicitly optimization statements that allow for a natural maximization of the probability of probabilistic worlds. Finally, note that all the available systems that can perform MPE for PASP (including those handling syntactic fragments of the formalism) are described and empirically compared in Section 5.

7 Conclusion

In this paper, we proposed three alternative techniques for solving the MPE problem in PASP. We start from an approach based on knowledge compilation and second-level algebraic model counting. Then, we proposed two logic-based encodings, one in ASP for the upper MPE state, and one that leverages ASP(Q), specifically targeting the lower MPE state. A broad experimental evaluation shows that ASP(Q) + ASP are the most effective solutions to solve MPE on non-stratified programs. There are many directions for future work, such as studying the considered tasks under different semantics and speeding up even more the execution time adopting multi-shot [Gebser *et al.*, 2019] approaches.

Acknowledgements

This work has been partially supported by Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU), by the Italian Ministry of Industrial Development (MISE) under project EI-TWIN n. F/310168/05/X56 CUP B29J24000680005, and by the Italian Ministry of Research (MUR) under projects: PNRR FAIR - Spoke 9 - WP 9.1 CUP H23C22000860006, Tech4You CUP H23C22000370006, and PRIN PINPOINT CUP H23C22000280006. DA and FR are members of the Gruppo Nazionale Calcolo Scientifico – Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

[Amendola *et al.*, 2019] Giovanni Amendola, Francesco Ricca, and Miroslaw Truszczynski. Beyond NP: quantifying over answer sets. *Theory and Practice of Logic Programming*, 19(5-6):705–721, 2019.

[Andres *et al.*, 2012] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *ICLP (Technical Communications)*, volume 17 of *LIPIcs*, pages 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[Azzolini and Riguzzi, 2023a] Damiano Azzolini and Fabrizio Riguzzi. Inference in probabilistic answer set programming under the credal semantics. In Roberto Basili, Domenico Lembo, Carla Limongelli, and Andrea Orlandini, editors, *AIxIA 2023 - Advances in Artificial Intelligence*, volume 14318 of *Lecture Notes in Artificial Intelligence*, pages 367–380, Heidelberg, Germany, 2023. Springer.

[Azzolini and Riguzzi, 2023b] Damiano Azzolini and Fabrizio Riguzzi. Lifted inference for statistical statements in probabilistic answer set programming. *International Journal of Approximate Reasoning*, 163:109040, 2023.

[Azzolini *et al.*, 2022] Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi. Statistical statements in probabilistic logic programming. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 43–55, Cham, 2022. Springer International Publishing.

[Azzolini *et al.*, 2023] Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi. MAP inference in probabilistic answer set programs. In Agostino Dovier, Angelo Montanari, and Andrea Orlandini, editors, *AIxIA 2022 – Advances in Artificial Intelligence*, pages 413–426, Cham, 2023. Springer International Publishing.

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.

[Bellodi *et al.*, 2020] Elena Bellodi, Marco Alberti, Fabrizio Riguzzi, and Riccardo Zese. MAP inference for probabilistic logic programming. *Theory and Practice of Logic Programming*, 20(5):641–655, 2020.

[Bogaerts *et al.*, 2016] Bart Bogaerts, Tomi Janhunen, and Shahab Tasharrofi. Stable-unstable semantics: Beyond NP with normal logic programs. *Theory and Practice of Logic Programming*, 16(5-6):570–586, 2016.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Buccafurri *et al.*, 2000] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860, 2000.

[Calimeri *et al.*, 2020] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca, and Torsten Schaub. ASP-Core-2 input language format. *Theory and Practice of Logic Programming*, 20(2):294–309, 2020.

[Cozman and Mauá, 2020] Fábio Gagliardi Cozman and Denis Deratani Mauá. The joy of probabilistic answer set programming: Semantics, complexity, expressivity, inference. *International Journal of Approximate Reasoning*, 125:218–239, 2020.

[Dantsin *et al.*, 2001] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[d'Avila Garcez *et al.*, 2019] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–631, 2019.

[De Raedt *et al.*, 2007] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In Manuela M. Veloso, editor, *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, volume 7, pages 2462–2467. AAAI Press, 2007.

[Eiter and Gottlob, 1995] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[Eiter *et al.*, 2021] Thomas Eiter, Markus Hecher, and Rafael Kiesel. Treewidth-aware cycle breaking for algebraic answer set counting. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 269–279, 2021.

[Eiter *et al.*, 2024] Thomas Eiter, Markus Hecher, and Rafael Kiesel. aspmc: New frontiers of algebraic answer set counting. *Artificial Intelligence*, 330:104109, 2024.

[Fandinno *et al.*, 2021] Jorge Fandinno, François Laferrière, Javier Romero, Torsten Schaub, and Tran Cao Son. Planning with incomplete information in quantified answer set programming. *Theory and Practice of Logic Programming*, 21(5):663–679, 2021.

[Gebser *et al.*, 2009] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Solution enumeration for projected boolean search problems. In Willem-Jan van Hoeve and John N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 71–86, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Gebser *et al.*, 2019] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, 2019.

[Geh *et al.*, 2024] Renato Lui Geh, Jonas Gonçalves, Igor C Silveira, Denis D Mauá, and Fabio G Cozman. dPASP: a probabilistic logic programming environment for neurosymbolic learning and reasoning. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, pages 731–742, 2024.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[Gondran and Minoux, 2008] Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms (Operations Research/Computer Science Interfaces Series)*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[Hahn *et al.*, 2024] Susana Hahn, Tomi Janhunen, Roland Kaminski, Javier Romero, Nicolas Ruhling, and Torsten Schaub. Plingo: A system for probabilistic reasoning in answer set programming. *Theory and Practice of Logic Programming*, page 1–34, 2024.

[Kiesel *et al.*, 2022] Rafael Kiesel, Pietro Totis, and Angelika Kimmig. Efficient knowledge compilation beyond weighted model counting. *Theory and Practice of Logic Programming*, 22(4):505–522, 2022.

[Kimmig *et al.*, 2017] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *Journal of Applied Logic*, 22(C):46–62, 2017.

[Lee and Wang, 2016] Joohyung Lee and Yi Wang. Weighted rules under the stable model semantics. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 145–154. AAAI Press, 2016.

[Lee and Yang, 2017] Joohyung Lee and Zhun Yang. LPMLN, weak constraints, and P-log. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1170–1177. AAAI Press, 2017.

[Mauá and Cozman, 2020] Denis Deratani Mauá and Fábio Gagliardi Cozman. Complexity results for probabilistic answer set programming. *International Journal of Approximate Reasoning*, 118:133–154, 2020.

[Mauá *et al.*, 2024] Denis Deratani Mauá, Fabio Gagliardi Cozman, and Alexandro Garces. Probabilistic logic programming under the l-stable semantics. In Nina Gierasimczuk and Jesse Heyninck, editors, *Proceedings of the 22nd International Workshop on Nonmonotonic Reasoning (NMR 2024)*, pages 24–33, 2024.

[Mazzotta *et al.*, 2024] Giuseppe Mazzotta, Francesco Ricca, and Mirek Truszczynski. Quantifying over optimum answer sets. *Theory and Practice of Logic Programming*, 23(4):948–964, 2024.

[Raedt *et al.*, 2016] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016.

[Rocha and Gagliardi Cozman, 2022] Victor Hugo Nascimento Rocha and Fabio Gagliardi Cozman. A Credal Least Undefined Stable Semantics for Probabilistic Logic Programs and Probabilistic Argumentation. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pages 309–319, 8 2022.

[Shterionov *et al.*, 2015] Dimitar Sht. Shterionov, Joris Renkens, Jonas Vlasselaer, Angelika Kimmig, Wannes Meert, and Gerda Janssens. The most probable explanation for probabilistic logic programs with annotated disjunctions. In Jesse Davis and Jan Ramon, editors, *24th International Conference on Inductive Logic Programming (ILP 2014)*, volume 9046 of *Lecture Notes in Computer Science*, pages 139–153, Berlin, Heidelberg, 2015. Springer.

[Stockmeyer, 1976] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[Totis *et al.*, 2023] Pietro Totis, Luc De Raedt, and Angelika Kimmig. smProbLog: Stable model semantics in ProbLog for probabilistic argumentation. *Theory and Practice of Logic Programming*, 23(6):1198–1247, 2023.