Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Advancing Algorithmic Approaches to Probabilistic Argumentation under the Constellation Approach

Andrei Popescu , Johannes P. Wallner
Institute of Software Technology, Graz University of Technology, Austria
{andrei.popescu, wallner}@ist.tugraz.at

Abstract

Reasoning with defeasible and conflicting knowledge in an argumentative form is a key research field in computational argumentation. Reasoning under various forms of uncertainty is both a key feature and a challenging barrier for automated argumentative reasoning. It was shown that argumentative reasoning using probabilities faces in general high computational complexity, in particular for the so-called constellation approach. In this paper, we develop an algorithmic approach to overcome this obstacle. We refine existing complexity results and show that two main reasoning tasks, that of computing the probability of a given set being an extension and an argument being acceptable, diverge in their complexity: the former is $\#\mathsf{P}$-complete and the latter is $\# \cdot \mathsf{NP}$-complete when considering their underlying counting problems. We present an algorithm for the complex task of computing the probability of a set of arguments being a complete extension by using dynamic programming operating on tree-decompositions. An experimental evaluation shows promise of our approach.

1 Introduction

The field of computational argumentation is nowadays a cornerstone of approaches to automated and rational argumentative reasoning within Artificial Intelligence (AI) (Baroni et al. 2018; Gabbay et al. 2021). Application avenues for this field include legal reasoning (Prakken and Sartor 2015), medical applications (Cyras et al. 2021; Fox et al. 2007), and multi-agent systems (Fan et al. 2014), see, e.g., the overview given by Atkinson et al. (2017).

Central to computational argumentation are formal approaches that define how reasoning is carried out. Common to many forms of argumentative reasoning is the utilization of argumentation frameworks (AFs) (Dung 1995), in which arguments are represented as vertices, and directed edges among arguments represent a directed conflict, or attack, relation. Importantly, it oftentimes suffices to abstract the internal structure of arguments, in order to find acceptable (sets of) arguments (Besnard et al. 2014).

Argumentation semantics specify which arguments can be deemed acceptable, and several such semantics exist for different purposes (Baroni, Caminada, and Giacomin 2011). A prominent property of such semantics is that of admissibility. A set of arguments is admissible if there are no attacks between two arguments in the set (i.e., they are conflict-free), and for each attack from outside the set onto the set there is a counter-attack from within, defending each argument in the set. An admissible set containing all arguments that are defended is called a complete extension.

Towards offering advanced forms of argumentative reasoning, AFs have been extended in several directions (Brewka, Polberg, and Woltran 2014). Recently, approaches that incorporate forms of uncertainty gained traction in research, e.g., by incorporating weights (Dunne et al. 2011) or allowing for forms of incompleteness (Baumeister et al. 2018). In probabilistic argumentation (Hunter et al. 2021; Hunter and Thimm 2017), uncertainty is captured by probabilities of, e.g., arguments and attacks, making them uncertain. Two main approaches to formalizing probabilities in AFs are the epistemic approach (Hunter 2013) and the constellation approach (Li, Oren, and Norman 2011).

In the constellation approach, subframeworks of a given AF constitute possible scenarios, thereby including only a part of the given AF. Each such subframework is associated with a probability, stating how probable this particular subframework (argumentative scenario) is. Two main reasoning tasks are then to compute the probability that a set is an extension under a specified semantics, such as complete semantics, and (credulous) acceptability of an argument. The former is defined as the sum of probabilities of subframeworks where the set is an extension, and the latter as the sum of probabilities of subframeworks where there is an extension containing the specified argument.

Complexity of probabilistic reasoning under the constellation approach was analyzed in-depth (Fazzinga, Flesca, and Parisi 2015; Fazzinga, Flesca, and Furfaro 2018; Fazzinga, Flesca, and Furfaro 2019; Alfano et al. 2023). The above two problems are $\mathsf{FP}^{\#\mathsf{P}}$-complete, e.g., for the complete semantics and when the probabilities are compactly represented as marginal probabilities of arguments and attacks with independence assumptions. Intuitively, the complexity results indicate that it is very challenging to compute the results: $\#\mathsf{P}$ (or $\mathsf{FP}^{\#\mathsf{P}}$) hard problems are presumed to be very difficult problems to solve. For instance, the archetypical $\#\mathsf{P}$-complete problem is that of counting satisfying truth-value assignments of a Boolean formula (Gomes, Sabharwal, and Selman 2021). Counting and probabilities are indeed connected, e.g., if each subframework has the same probability, the reasoning tasks above amount to counting the subframeworks satisfying the chosen criteria.

The high computational complexity was likely so far a major barrier for development of algorithmic approaches and systems for solving tasks in probabilistic argumentation. To the best of our knowledge, there are not many algorithms and systems for the constellation approach for AFs for the computationally hard tasks. As an exception, Fazzinga et al. (2019) provide algorithms for bipolar frameworks (Amgoud et al. 2008), yet for probabilistic AFs their results do not apply to $\mathsf{FP}^{\#\mathsf{P}}$-complete reasoning tasks. Moreover, Alfano et al. (2023) present an approximation algorithm for a variant of acceptability of an argument. In this paper we take up this challenge and develop an exact algorithmic approach to probabilistic argumentation frameworks under the constellation approach.

Our main contributions are as follows.

- We first refine existing complexity results stating that computing the probability of a set being an extension and acceptability share the same complexity. We show, using counting complexity classes, that their underlying counting problems differ in their complexity: the former problem is $\#\mathsf{P}$-complete, for the complete semantics, and the latter is $\# \cdot \mathsf{NP}$-complete for admissible, complete, and stable semantics, indicating a jump in the counting complexity hierarchy. Moreover, even when restricting to acyclic attack structures, the latter problem remains $\#\mathsf{P}$-complete.
- Next we look into algorithms to solve these tasks. Towards efficient reasoning, we give results in preprocessing probabilistic AFs to simplify given instances.
- Inspired by their capability of solving $\#\mathsf{P}$-hard problems, we develop a dynamic programming algorithm for probabilistic AFs utilizing tree-decompositions (Bodlaender 1993). Our algorithm is capable of solving the $\mathsf{FP}^{\#\mathsf{P}}$-complete problem of computing the probability of a set being a complete extension.
- We experimentally evaluate a prototype of our algorithm, which is publicly available under an open license, and show promise of our approach that solves PAFs up to 750 arguments, depending on the attack-structure.
- Finally, we discuss extensions of our approach, e.g., to incorporate dependencies among arguments, relaxing independence assumptions.

2 Background

We recall main definitions of argumentation frameworks (AFs) (Dung 1995) and probabilistic argumentation (Hunter et al. 2021; Hunter and Thimm 2017) under the constellation approach (Li, Oren, and Norman 2011). Moreover, we recap the notion of tree-width and tree-decompositions (Bodlaender 1993) required for our work.

Definition 1. *An argumentation framework (AF) is a pair (A, R) where A is a finite set of arguments and $R \subseteq A \times A$ is an attack relation.*

For a given AF $F = (A, R)$, if $(a, b) \in R$, then a attacks b (in F). Similarly, a set $S \subseteq A$ attacks $b \in A$ if there is an $a \in S$ that attacks b (in F).

For specifying possible AFs, we make use of the notion of subframeworks. A subframework of an AF $F = (A, R)$ is an AF $F' = (A', R')$ with $A' \subseteq A$ and $R' \subseteq R$. Note that F' is an AF, thus $R' \subseteq A' \times A'$ holds. The set of all subframeworks is denoted by $\mathcal{F}(F) = \{F' \mid F' \text{ a subframework of } F\}$.

Argumentation semantics (Dung 1995; Baroni, Caminada, and Giacomin 2011) on AFs are defined via functions $\sigma(F)$, for a given AF $F = (A, R)$, that assign subsets of the arguments as σ-extensions, i.e., $\sigma(F) \subseteq 2^A$. Central to AF semantics is the notion of defense.

Definition 2. *Let $F = (A, R)$ be an AF. A set of arguments $S \subseteq A$ defends an argument $a \in A$ if it holds that whenever $(b, a) \in R$ there is a $c \in S$ with $(c, b) \in R$.*

We next define main semantics on AFs.

Definition 3. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is* conflict-free (in F)*, if there are no $a, b \in S$, s.t. $(a, b) \in R$. We denote the collection of conflict-free sets of F by $cf(F)$. For a conflict-free set $S \in cf(F)$, it holds that*

- *$S \in stb(F)$ iff S attacks each $a \in A \setminus S$;*
- *$S \in adm(F)$ iff S defends each $a \in S$;*
- *$S \in com(F)$ iff $S \in adm(F)$ and whenever $a \in A$ is defended by S, then $a \in S$; and*
- *$S \in grd(F)$ iff $S \in com(F)$ and there is no $T \in com(F)$ with $T \subsetneq S$.*

We refer to subsets of arguments that are in $\sigma(F)$ as σ-extensions and also as an extension under a semantics σ, for $\sigma \in \{stb, adm, com, grd\}$.

For developing our algorithms later on it will be useful to also consider the labeling-based definitions of the argumentation semantics (Caminada and Gabbay 2009).

Definition 4. *Let $F = (A, R)$ be an AF. A labeling $L : A \to \{I, O, U\}$ in F is a function assigning a label to each argument in F.*

Intuitively, I ("in") signals accepting the argument in a labeling, O ("out") is interpreted as attacked by accepted arguments, and U ("undecided") takes neither stance. We sometimes use partial labelings, which are labelings where L is partial. To distinguish extensions and labelings under a semantics, we use σ and σ_{Lab}, respectively, unless clear from the context. We sometimes view labelings as triples, arguments assigned I, O, and U, respectively.

Definition 5. *Let $F = (A, R)$ be an AF. For a labeling L in F it holds that*

- *$L \in adm_{Lab}(F)$ iff for each $a \in A$ we find that*
  - *$L(a) = I$ implies that $L(b) = O$ if $(b, a) \in R$ and*
  - *$L(a) = O$ implies $\exists (b, a) \in R$ with $L(b) = I$,*
- *$L \in com_{Lab}(F)$ iff $L \in adm_{Lab}(F)$ and for $a \in A$ $L(a) = U$ implies $\exists (b, a) \in R$ with $L(b) = U$ and $\nexists (b, a) \in R$ with $L(b) = I$, and*
- *$L \in stb_{Lab}(F)$ iff for all $a \in A$ it holds that $L(a) \neq U$ and $L \in adm_{Lab}(F)$.*

There is a direct correspondence between the extension-based and labeling-based view. For a given AF $F = (A, R)$

and $S \in cf(F)$, the corresponding labeling L_S is defined for $a \in A$ by $L_S(a) = I$ if $a \in S$, $L_S(a) = O$ if $\exists (b, a) \in R$ with $b \in S$, and $L_S(a) = U$ if $a \notin S$ and $\nexists (b, a) \in R$ s.t. $b \in S$. Conversely, for a given labeling L in F, the corresponding extension S_L is defined as $\{a \mid L(a) = I\}$. We recall the main correspondence results (Caminada and Gabbay 2009, Theorem 9 and Theorem 10).

Proposition 1. *Let $F = (A, R)$ be an AF and $\sigma \in \{com, stb\}$.*

- *If $S \in \sigma(F)$ then $L_S \in \sigma_{Lab}(F)$ and*
- *if $L \in \sigma_{Lab}(F)$ then $S_L \in \sigma(F)$.*

Example 1. *Consider an AF $F = (A, R)$ with $A = \{a, b, c, d, e\}$ and attacks $R = \{(a, b), (a, d), (b, a), (b, c), (c, b), (c, d), (d, c), (d, a), (d, e), (e, d)\}$, see also Figure 1 (we explain the difference between solid and dashed components later). In this AF, the complete extensions are given by $com(F) = \{\emptyset, \{e\}, \{b\}, \{b, e\}, \{b, d\}, \{a, c, e\}\}$. For instance, set $E = \{a, c, e\}$ is complete since there are no attacks among these arguments (conflict-freeness), all attacks outside are countered (admissibility) and all defended arguments are included. A corresponding complete labeling is $L_E = \{a \mapsto I, b \mapsto O, c \mapsto I, d \mapsto O, e \mapsto I\}$ which is also a stable labeling.*

We move on to probabilistic argumentation frameworks under the constellation approach (Li, Oren, and Norman 2011). A probability distribution function (pdf) for an AF F is a function $P : \mathcal{F}(F) \to [0, 1]$ s.t. $\sum_{F' \in \mathcal{F}(F)} P(F) = 1$.

Definition 6. *A probabilistic argumentation framework (PAF) is a triple (A, R, P) where (A, R) is an AF and P is a pdf over $\mathcal{F}(F)$.*

For the main part of the paper, we follow the approach dubbed "IND", where P is defined via the marginal probabilities of arguments and attacks, by assuming independence, as defined by (Li, Oren, and Norman 2011; Hunter 2013; Fazzinga, Flesca, and Furfaro 2019). We discuss extensions in Section 7. That is, given a PAF $F = (A, R, P)$, $P : A \cup R \to [0, 1]$ assigns probabilities to arguments and attacks. For each $F' = (A', R') \in \mathcal{F}(F)$ the probability of the subframework F' is $P(F')$ defined as

$$\prod_{a \in A'} P(a) \cdot \prod_{a \in A \setminus A'} 1 - P(a) \cdot \prod_{r \in R'} P(r) \cdot \prod_{r \in D \setminus R'} 1 - P(r)$$

with $D = R \cap (A' \times A')$ the set of all attacks in R between arguments in A'. That is, the probability of F' is the product of $P(a)$ if $a \in A'$ is present, $1 - P(a)$ whenever a was "removed" from A (is in $A \setminus A'$), and for each attack r, we use $P(r)$ if the attack is present in F' and use $1 - P(r)$ only if the attack was removed, but could be present (i.e., both endpoints are in A').

By definition, $P(x) = 0$ indicates that subframeworks with x (either argument or attack) have 0 probability. W.l.o.g., we assume that all arguments and attacks with zero probability are removed from a given PAF F. For arguments and attacks that are certain (i.e., P assigns probability one), we can restrict our focus to only those subframeworks that contain a certain argument and a certain attack if both endpoints of the attack are present. Towards this, we extend

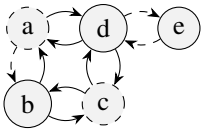

Figure 1: A PAF with certain (solid lines) and uncertain (dashed lines) arguments and attacks.

the notion of subframeworks via $\mathcal{F}_P(F) = \{(A', R') \in \mathcal{F}(F) \mid \forall a \in A, P(a) = 1$ implies $a \in A'$ and $\forall (a, b) \in R, \{a, b\} \subseteq A', P((a, b)) = 1$ implies$(a, b) \in R'\}$. That is, all subframeworks in $\mathcal{F}_P(F)$ agree on the certain part (for attacks if both endpoints are there).

Example 2. *Let us continue Example 1 and include probabilities to the AF $F = (A, R)$ (Figure 1). Let $F' = (A, R, P)$ be a PAF, with $P(a) = 0.8$, $P(c) = 0.9$, $P((a, b)) = 0.7$, $P((d, e)) = 0.5$, $P((e, d)) = 0.3$, and all other probabilities equal to 1 (solid lines in the figure). This PAF gives rise to 24 subframeworks in $\mathcal{F}_P(F')$. We list all subframeworks in the supplement. Consider the subframework $F'' = (\{b, c, d, e\}, \{(b, c), (c, b), (d, c), (c, d), (e, d)\}$. We have $P(F'') = P(b) \cdot P(c) \cdot P(d) \cdot P(e) \cdot (1 - P(a)) \cdot P((b, c)) \cdot P((c, b)) \cdot P((d, c)) \cdot P((c, d)) \cdot P((e, d)) \cdot (1 - P((d, e)) = 1 \cdot 0.9 \cdot 1 \cdot 0.2 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 0.3 \cdot 0.5 = 0.027$.*

The reasoning tasks we focus on are computing the probability of a set of arguments being an extension (P-Ext) and the probability of an argument being credulously accepted (P-Acc) (Fazzinga, Flesca, and Furfaro 2019). For conciseness, we shorten the latter and simply say "accepted".

Definition 7. *Let $F = (A, R, P)$ be a PAF, $\sigma \in \{adm, com, stb\}$ a semantics, and $a \in A$ an argument. The probability that $S \subseteq A$ is a σ-extension (in F) is*

$$P^{ext}_\sigma(S) = \sum_{F' \in \mathcal{F}(F), S \in \sigma(F')} P(F')$$

and the probability of accepting a under σ (in F) is

$$P^{acc}_\sigma(a) = \sum_{F' \in \mathcal{F}(F), \exists S \in \sigma(F'), a \in S} P(F').$$

In words, the probability of a set of arguments being a σ-extension is defined as the sum of probabilities of subframeworks for which it holds that they have S as a σ-extension. Similarly, the probability that a is accepted under σ in F is the sum of probabilities of subframeworks where there is a σ-extension containing the argument.

Example 3. *Continuing Example 2, let $S = \{a, c, e\}$, and the query argument be e. We obtain $P^{ext}_{com}(\{a, c, e\}) = 0.72$ by summing the probabilities of all subframeworks in which S is a complete extension. Similarly we have $P^{acc}_{com}(e) = 0.98$ by summing the probabilities of all subframeworks in which there exists a complete extension with argument e.*

We recall main complexity results. For background on complexity, we refer the reader to standard textbooks on complexity (Papadimitriou 2007). Important for probabilistic reasoning are counting complexity classes. A major complexity class for counting is $\#\mathsf{P}$ that contains all

functions that count the number of accepting paths of a non-deterministic polynomial-time Turing machine (Valiant 1979). It was shown (Fazzinga, Flesca, and Parisi 2015; Fazzinga, Flesca, and Furfaro 2018) that it is $\mathsf{FP}^{\#\mathsf{P}}$-complete to compute the probability of a set of arguments being a complete extension and the problem of computing the probability of an argument being accepted under admissibility, complete and stable semantics, has the same complexity. The probability of a set of arguments being admissible or stable, on the other hand, can be computed in polynomial time.

Tree-Decompositions and Tree-Width Intuitively, tree-width measures a distance of an undirected graph $G = (V, E)$ to being a tree. A tree-decomposition of a graph G can be, on the one hand, used for measuring tree-width and, on the other hand, is a useful data structure for computation. For a given PAF $F = (A, R, P)$, we associate an undirected graph $F_G = (V, E)$ with F in a direct way: $V = A$ and $E = R$, but we interpret E as undirected edges.

For a given tree $T = (V, E)$, we define the shorthand $T_N = V$ for the nodes in T. A tree-decomposition of an undirected graph $G = (V, E)$ is a pair $(T, (B_t)_{t \in T_N})$, with T being a rooted tree and each $B_t \subseteq V$ such that the following properties are satisfied. A B_t is also called a "bag" (containing vertices of G). Every vertex $x \in V$ is part of a bag B_t, i.e., $x \in B_t$ for some $t \in T_N$. For every $(x, y) \in E$ there is a bag B_t, $t \in T_N$, such that $\{x, y\} \subseteq B_t$. The set $\{t \mid x \in B_t\}$ induces a subtree of T, for each $x \in V$. In brief terms, a tree-decomposition is a tree such that its nodes (the bags) are connected and all vertices of the original graph G are contained in some bag. The second condition ensures that each edge is present in at least one bag. The third condition, often referred to as the connectedness condition, states that whenever two bags B_t and $B_{t'}$ contain a vertex x, then on the path between those two bags, we encounter x in all the bags.

The width of a tree-decomposition $(T, (B_t)_{t \in T_N})$ is the maximum number of vertices in bags minus one, i.e., $max\{|B_t| \mid t \in T_N\} - 1$. The tree-width of an undirected graph $G = (V, E)$ is the minimum width of all tree-decompositions of G. We give illustrations of tree-decompositions in Section 5.

3 Complexity Results for Probabilistic AFs

We investigate the complexity of probabilistic reasoning. In particular, we present novel results for counting variants of probabilistic reasoning.

We refine existing results, and show that the complexity of problems of computing the probability of a set of arguments and acceptability of an argument diverges on their underlying counting problems. Towards our result, we also consider the counting complexity classes $\# \cdot \mathsf{P}$ and $\# \cdot \mathsf{NP}$. These "dot" classes are from a hierarchy of counting complexity classes (Hemaspaandra and Vollmer 1995), and are defined as follows. A counting problem is defined via a witness function $w : \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$ that assigns, given a string from alphabet Σ a collection of (finite) subsets from alphabet Γ. The task is to count $|w(x)|$, given x. Additionally, we require that each witness $y \in w(x)$ is polynomially bounded by x. A counting problem is in $\# \cdot \mathcal{C}$, for a class of decision problems $\mathcal{C}$, if given x and y the problem to decide whether $y \in w(x)$ is in $\mathcal{C}$. For illustration, the archetypical $\# \cdot \mathsf{P}$-complete (as for $\#\mathsf{P}$) is $\#\mathsf{SAT}$, with the witness function assigning satisfying truth-value assignments to a formula. It holds that $\#\mathsf{P} = \# \cdot \mathsf{P}$, i.e., we can interchangeably use these two classes.

Let us consider the following counting problems, for a given PAF $F = (A, R, P)$ and semantics σ:

- given a set of arguments $S \subseteq A$, count the number of subframeworks $F' \in \mathcal{F}_P(F)$ for which we find that S is a σ-extension in F', and
- given an argument $a \in A$, in how many subframeworks $F' \in \mathcal{F}_P(F)$ is it the case that there is an $S \in \sigma(F')$ with $a \in S$?

Example 4. *Let $F = (A, R, P)$ be a PAF with all attacks being certain and all arguments having probability 0.5. By definition, for any $F' \in \mathcal{F}(F)$ it holds that $P(F') = \prod_{a \in A'} 0.5 \cdot \prod_{a \in A \setminus A'} (1 - 0.5) = 0.5^{|A|}$. If there are n many subframeworks for which a specific $S \subseteq A$ is a σ-extension, then the probability of S being a σ-extension in F is $n \cdot 0.5^{|A|}$, which is fully determined by n.*

As the example suggests, counting the number of subframeworks satisfying the above criteria acts as special case of computing probabilities of a set being an extension or of the acceptability of an argument. In the general case, as recently studied in more detail (Fazzinga et al. 2024), the counting variants and the probabilistic problems are not in a very direct relation to each other, from a computational viewpoint.

We start with the problem of counting the number of subframeworks where a given set is an extension. While we will show $\#\mathsf{P}$-completeness ($\# \cdot \mathsf{P}$-completeness), we first argue that this problem is not $\#\mathsf{P}$-hard under parsimonious reductions. The type of reductions are important for counting complexity classes, since closure of counting classes under some types of reductions is not immediate (Durand, Hermann, and Kolaitis 2005). A reduction is parsimonious if a counting problem is transformed to another in polynomial time and the number of solutions (cardinality of their respective witness sets) is preserved exactly.

By a result of Fazzinga, Flesca, and Furfaro, Theorem 2 (2020), it holds that one can check in polynomial time whether *some* subframework has a queried set as a complete extension. This directly prevents existence of a parsimonious reduction in our case, under complexity theoretic assumptions. If there were a parsimonious reduction from $\#\mathsf{SAT}$ to this problem, one could solve SAT in polynomial time: one can reduce a Boolean formula to the current problem and in polynomial time find one suitable subframework, which in turn implies that there is a satisfying truth value assignment of the formula. This gives rise to the following result.

Proposition 2. *Unless $\mathsf{P} = \mathsf{NP}$, there is no parsimonious reduction from $\#\mathsf{SAT}$ to counting the number of subframe-*

works of a given PAF that have a given set of arguments as a complete extension.

Nevertheless, we show hardness for a more relaxed notion, Turing reductions, in which a counting problem A is reduced to B by A being solvable in polynomial time, given that we can use B as an oracle. We remark that, as discussed by Durand, Hermann, and Kolaitis (2005), for $\#\mathsf{P}$ it is unclear whether closure under Turing reductions holds. We conjecture that other suitable types of reductions where closure was proven (like subtractive reductions proposed by Durand, Hermann, and Kolaitis, 2005) can be applied here. In any case, membership in $\#\mathsf{P}$ shows the result for diverging complexity we aim for.

Theorem 3. *It is $\#\mathsf{P}$-complete, under Turing reductions, to count the number of subframeworks where a given set of arguments is a complete extension in probabilistic AFs.*

For the problem of counting the number of subframeworks where a given argument is accepted, under a semantics, we show $\# \cdot \mathsf{NP}$-completeness, under parsimonious reductions, for the main semantics considered in this work.

Theorem 4. *It is $\# \cdot \mathsf{NP}$-complete, under parsimonious reductions, to count the number of subframeworks where a given argument is accepted in probabilistic AFs under stable, complete, and admissible semantics.*

In our reduction one part of the constructed PAF is, in fact, acyclic, i.e., the directed graph (A, R) in the PAF is acyclic. This leads to the following result.

Theorem 5. *It is $\# \cdot \mathsf{P}$-complete, under parsimonious reductions, to count the number of subframeworks where a given argument is accepted in probabilistic acyclic AFs under grounded, stable, complete, and admissible semantics.*

That is, even when restricted to an acyclic attack structure, $\#\mathsf{P}$-hardness persists, which is in stark contrast to other abstract argumentation formalisms (Dvořák and Dunne 2018).

We next delve into positive complexity results. We show "fixed-parameter tractability", under the parameter tree-width.

An algorithm is a fixed-parameter algorithm (Downey and Fellows 1999) w.r.t. parameter k for a given problem, if the algorithm solves a problem in time $\mathcal{O}(f(k) \cdot n^c)$, with n the size of an instance, c a constant, k a non-negative integer k, and f a computable function. Intuitively, the running time may depend exponentially on the parameter k and polynomially on the instance size. Our algorithm to be presented in Section 5 witnesses fixed-parameter tractability, under the parameter tree-width. In particular, the data structures generated are bounded by the size of bags (i.e., tree-width).

Theorem 6. *There is a fixed-parameter algorithm, w.r.t. the parameter tree-width, for the problem of computing the probability of a set of arguments being a complete extension.*

4 Preprocessing Probabilistic AFs

Here we look at preprocessing for PAFs. In general, preprocessing is seen as a vital component for solving hard problems, e.g., in SAT solving (Biere, Järvisalo, and Kiesl 2021) to simplify instances.

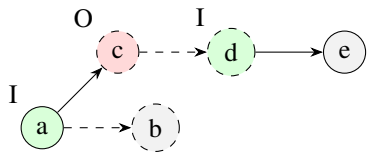

Figure 2: Preprocessing a PAF

Example 5. *Consider the PAF shown in Figure 2 with solid components certain and dashed ones uncertain. Since argument a is unattacked and certain, all subframeworks in $\mathcal{F}_P(F)$ contain a and in each complete labeling a is labeled "in". Argument b is labeled uncertain even if attacked by a certain, unattacked argument, since the attack itself is uncertain. That is, there is the possibility of having a subframework with b labeled I in a complete labeling (those without the attack (a, b)). On the other hand, argument c can safely by labeled "out", whenever c is in a subframework: then also a is present and both attacker and attack are present. Argument d is uncertain, but can safely be labeled I in each subframework: in case there is the attacker c and attack (c, d), then c will be labeled O. Finally, argument e is attacked by d, if d is present, and d is I, however, d might not be there at all. Thus there is no fixed stance towards e.*

Formally, let us define $\mathcal{A}(F)$, for a given PAF $F = (A, R, P)$ and a given partial labeling L, that returns a partial labeling L' as follows. $L'(a)$ is I if for all attacks of (b, a) in F it holds that $L(b) = O$. $L'(a)$ is O if there is an attack (b, a) in F such that $P(b) = 1$, $P(b, a) = 1$ and $L(b) = I$. We have to label an argument "in" if the argument is present in a subframework and for all attacks (if existing) it is clear that they are "defeated". An argument is "out" if a certain argument labeled I and attack designate it to be out. Let L be the least-fixed point of this function. It holds that for $P\text{-}Ext$ and a given set S, the probability is 0 if S contains an argument labeled "out" in L. Moreover, if an argument $a \notin S$ is labeled "in" by L, then (i) if a is certain, the probability is 0 and (ii) if a is uncertain only subframeworks without a can have S as a complete extension. For $P\text{-}Acc$ an argument a labeled "out" in L has 0 probability.

5 Tree-Decomposition-based Algorithm

In this section we develop our algorithm for computing the probabilities of a given set being an extension ($P\text{-}Ext$). We illustrate our algorithms mainly on the complete semantics, yet our algorithms can be utilized also for admissible sets and stable semantics. Nevertheless, under complete semantics the considered problem is $\mathsf{FP}^{\#\mathsf{P}}$-complete.

Our algorithm operates on tree-decompositions, more specifically so-called nice tree-decompositions (Bodlaender and Koster 2008), for a given PAF $F = (A, R, P)$. A nice tree-decomposition is a tree-decomposition where, additionally, we find that each node is one of the following types:

- the root node or a leaf node with empty bags,
- a join node, which is a node t with exactly two children, t_1 and t_2, such that $B_t = B_{t_1} = B_{t_2}$,

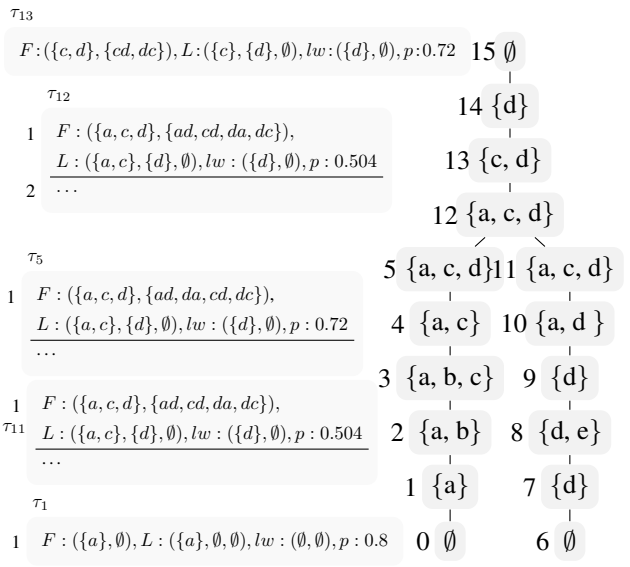

Figure 3: Nice tree decomposition of the PAF of Example 2 and corresponding tables (partial) for the computation of $P^{ext}_{com}(\{a, c, e\})$.

- an introduction node, which is a node t with exactly one child t' such that $B_t = B_{t'} \cup \{x\}$ and $x \in A$, or
- a forget node, which is a node t with exactly one child t' such that $B_t = B_{t'} \setminus \{x\}$ and $x \in A$.

The first condition (empty bags for the root and leaf nodes) is not in the original definition of nice tree-decompositions, but can be imposed directly and is useful for developing our algorithm. A join node has two children and all share the same bag, introduction and forget nodes each have one child and either introduce exactly one or forget exactly one argument. Given a (not nice) tree-decomposition, a nice tree-decomposition can be obtained in linear time (Kloks 1994).

Example 6. *The PAF from Example 2 can be represented by the nice tree-decomposition as shown in Figure 3. Here each node has a unique identifier, e.g., the root has the identifier 15. Node 1 introduces argument a, node 4 forgets argument b, and node 12 is a join node.*

For a constant k, there is a linear-time algorithm that checks whether a given undirected graph has tree-width k, and if reporting positively, also can return a tree-decomposition of minimum width (Bodlaender 1996). Practically, we employ a library (Abseher, Musliu, and Woltran 2017) using heuristics for the generation of a tree-decomposition.

The algorithm we develop performs a bottom-up computation on a nice tree-decomposition. Starting from the leaves, for each bag a table of rows is computed. Each row can be seen as a partial solution, applicable to the part of the PAF that was visited so far.

Algorithm 1 is the main algorithm, which calls sub algorithms depending on the type of the current node. In Line 1 we construct a nice tree-decomposition of the given PAF and initialize an empty table (set) for each bag in Line 2. We go over all bags in post-order in the loop in Line 3. After the case distinction for each type of nodes, we return the probability given in $(\emptyset, \emptyset, p)$ in the root of the tree-decomposition. There is only one such row in each computation.

For a given (input) PAF $F = (A, R, P)$, we make use of some auxiliary definitions and shorthands. The main datastructure of our algorithms is a row (s, w, p) in a table, which is composed of a structure s, a witness w, and a probability $p \in [0, 1]$. A structure $s = (F', L')$ is a pair of a subframework F' of $\mathcal{F}_P(F)$ and a labeling L' in F'. What we call a witness w contains a labeling-witness lw, which is a partial labeling in F'.

Intuitively, if a row (s, w, p) is computed in a table τ_t for node t and bag B_t, this means that the sum of probabilities of all "completions" of s to arguments in $B_{\leq t}$ that respect the witness w is equal to p. With $B_{\leq t} = \bigcup_{t' \in T'} B_{t'}$ we define the union of all bags in the subtree T' with root t. For instance, $B_{\leq 4}$ of Figure 3 is $\{a, b, c\}$. A completion of a structure s to $B_{\leq t}$ (all arguments "seen") are all completions of the current subframework to subframeworks containing arguments and attacks in $B_{\leq t}$ (and taking certain parts into account), as well as all labelings that extend the one in s up to $B_{\leq t}$. These labelings have to satisfy that those arguments in S that are in $B_{\leq t}$ (i.e., $S \cap B_{\leq t}$) are labeled I, and for all arguments already seen but not in the current bag ($B_t \setminus B_{\leq t}$) the conditions of being complete are satisfied. For the arguments inside the bag, whenever the label-witness assigns "out" or "undecided" to an argument, there must be a reason.

This means, if one extends the structure s in all possible ways up to $B_{\leq t}$ that still adhere to the conditions of subframeworks of PAFs and complete labelings for those arguments outside the bag, all these structures are "valid" subframeworks to sum. Since, by construction, the root node has an empty bag, and all arguments have been traversed before, all subframeworks have been considered and their labelings where S is complete (there is exactly one such labeling in each subframework), the result in the root node is the probability of S being a complete extension.

A witness w is used to "remember" facts that decide whether the row can be associated to a complete labeling.

We next make the above intuition more formal. Let $r = (s, w, p)$ be a row in τ_t for node t with $s = (F', L')$ and $w = lw$, for a given PAF $F = (A, R, P)$. Define that $F'' = (A'', R'')$ is an expansion of F' and B_t if $A' \subseteq A''$, $R' \subseteq R''$, $(B_t \setminus A') \cap A'' = \emptyset$, and no attacks possible via arguments in B_t that are not in R' are

Algorithm 1 P-Ext($F = (A, R, P), \sigma, S$)

```
1: Compute nice TD (T, (B_t)_{t∈T_N}) of (A, R)
2: Let τ_t := ∅ for each t ∈ T_N
3: for t ∈ T_N in post-order
4:     if t is a leaf set τ_t := {(∅, ∅, 1)}
5:     if t of type x ∈ {Intro, Forget} with child t'
6:         τ_t := x(τ_t', B_t, S)
7:     if t of type Join with children t_1, t_2
8:         τ_t := Join(τ_t1, τ_t2, B_t, S)
9: return p with (∅, ∅, p) ∈ τ_t with t the root
```

in R''. Let $F_{\leq t} = (A_{\leq t}, R_{\leq t}) \in \mathcal{F}_P(F)$ be the subframework s.t. $A_{\leq t} = B_{\leq t}$ and $R_{\leq t} = R \cap A_{\leq t}$. Then $\mathcal{F}_P^{\leq t}(F) = \{F'' \in \mathcal{F}_P(F_{\leq t}) \mid F''$ expands F' and $B_t\}$ is the set of all subframeworks with contents restricted to $B_{\leq t}$ expanding F'.

For L', its completion to $A'' \supseteq A'$ is a labeling that assigns the same labels to those in A' as L' and some labeling to those in $A'' \setminus A'$. A labeling L'' on a subframework F'' is said to be partially complete w.r.t. bag B_t, set S, and labeling-witness lw if for each argument a in F'' the following holds. If $a \in B_{\leq t} \setminus B_t$ then $L(a) = I$ implies all attackers in F'' are O, $L(a) = O$ implies there is an attacker in F'' that is I, and $L(a) = U$ implies no attacker is I and there is a U attacker. For arguments $a \in B_t$, we require that $L(a) = I$ implies that all adjacent arguments are O, $lw(a) = O$ iff there is an attacker that is I in F'', and $lw(a) = U$ iff there is a U attacker and no attacker is I. Finally each argument in S must be assigned "in". That is, the conditions for complete labelings is satisfied for arguments already "seen" but forgotten, and for arguments in the current bag, we only require (i) conflict-freeness and (ii) if the labeling-witness specifies an argument to be "out" or "undecided", there is a justification.

Let $\mathcal{X}_r$ be the set of pairs (F'', L'') such that $F'' \in \mathcal{F}_P^{\leq t}(F)$ and L'' is partially complete in F'' w.r.t. bag B_t, set S and lw. It holds that p is the sum of all subframeworks in $\mathcal{X}_r$ (restricted to components in $B_{\leq t}$). Intuitively, $\mathcal{X}_r$ contains all "compatible" subframeworks that respect the witness, the current labeling, and the queried set S, but relaxing the conditions for the arguments inside the bag, since adjacent arguments might not be traversed yet.

Introduction Nodes Algorithm 2 presents our approach to introduction nodes t. The child node of t is t' and τ' contains the rows for t'. Intuitively, for each row $r' = (s, w, p)$ from t' we compute new rows r, by either adding the introduced argument a or not to the subframework in s (Line 2 and Line 5). If the argument was added, then for each possible way of labeling the new argument (Line 3) we compute new labelings and update witnesses.

Given a PAF $F = (A, R, P)$, as a shorthand, we define $\mathcal{S}_{s+a}(F)$, for a structure $s = (F', L')$ with $F' = (A', R')$, an argument a, to be all structures that may extend s with a new argument a. We define the set of AFs extending F' by adding a and possibly incident attacks.

$$F'_{+a} = \{F'' = (A'', R'') \in \mathcal{F}_P(F) \mid A'' = A' \cup \{a\}, \\ R' \subseteq R'' \subseteq R' \cup (\{(a, x), (x, a) \mid x \in A''\} \cap R)\}$$

Then $\mathcal{S}_{s+a}(F)$ is equal to $\{(F'', L'') \mid F'' \in F'_{+a}, L''$ completion to $F''\}$. That is, $\mathcal{S}_{s+a}$ contains all pairs with an AF F'' and labeling L'' such that F'' additionally contains a, all arguments and attacks from F', and may add attacks of a (restricted to those originally in R).

When adding an argument a, the probability is updated by multiplying with $P(a)$ and considering the factors for each added attack in the structure ($Add(s, a)$) and attack from R not added ($Rem(s, a)$).

$$\texttt{UpP}(s, a, p) = p \cdot P(a) \cdot Atts(s, a)$$
$$Atts(s, a) = \prod_{r \in Add(s,a)} P(r) \cdot \prod_{r \in Rem(s,a)} (1 - P(r))$$

The witnesses are updated as follows: $\texttt{UpW}(s, w) = w \cup \{x \mapsto O \mid (y, x) \in R, L(y) = I\} \cup \{x \mapsto U \mid (y, x) \in R, L(y) = U\}$. In brief, we store whenever an argument can be assigned to "out" or "undecided": if there is an attacked argument that is "undecided" then its attacker must also be "undecided", as required for the complete semantics. An "in" attacker is taken care of by requiring in introduction nodes all adjacent arguments to be "out", as specified in the following.

The condition $\sigma_{intro}(s)$ ensures that the labelings in s are conflict-free, i.e., not adjacent arguments are labeled "in" and arguments adjacent to ones labeled I are labeled O. For a resulting set of rows τ, the function $\texttt{checkAcceptance}(\tau)$ filters those rows where S is not labeled I. In practice, this filter can be applied earlier (during row construction). For the sake of readability, we opted to show this filter here.

Example 7. *Consider introduction node 1 from Example 6 (Figure 3), and its table τ_1. The unique $r' \in \tau_0$ is the special case $(\emptyset, \emptyset, 1)$. There are two possibilities in terms of subframeworks: adding a or not. Since $a \in S$, the latter is filtered out. When adding a, we consider new labelings to construct. In this instance, only labeling a to be "in" is permitted, since $a \in S$. The probability is then $p = 1 \cdot 0.8 = 0.8$. Finally, the labeling-witness stays empty.*

Consider a row $r = (s, w, p)$ created in an introduction node. Then there was previously a row $r' = (s', w', p')$ from which r was constructed. If r' satisfies that p' is the sum of probabilities of subframeworks in $\mathcal{X}_{r'}$ (as defined above), then also p in r satisfies the same property: either a was not added (then it is direct that subframeworks in $\mathcal{X}_r$ are the same as in $\mathcal{X}_{r'}$ and we multiply with $(1 - P(a))$ and in the other case, we created a subframework with a, some incident attacks, and a labeling that satisfies conflict-freeness. In the latter case, since the bag only increased, the arguments in the bag are not yet fully checked for being complete.

Forget Nodes Let us move to forget nodes (Algorithm 3). The main idea here is that an argument a is "forgotten" in a bag (compared to the bag of the unique child in the nice tree-decomposition). This is the point in the algorithm where we can verify that the label assigned to a does not violate the conditions for being complete, since all adjacent arguments

Algorithm 2 Introduction(τ', B_t, S)

```
1: for (s, w, p) ∈ τ'
2:     if P(a) ≠ 1 then τ := τ ∪ {(s, w, p · (1 − P(a)))}
3:     for s' ∈ S_{s+a}(F)
4:         if σ_intro(s') then
5:             τ := τ ∪ {(s', UpW(s', w), UpP(s', a, p))}
6: return checkAcceptance(τ, S)
```

were traversed already. The witness-label lw for a enables us to check conditions for a.

Say we are in node t with child node t'. Consider a row from t' as $r' = (s', w', p')$ with $s' = (F', L')$. If we assume that p' is the sum of probabilities of subframeworks in $\mathcal{X}_{r'}$, then to compute a new row $r = (s, w, p)$, based on r', we need to make sure that for all subframeworks and labels associated with $\mathcal{X}_r$ we get that (i) the now forgotten argument satisfies the complete conditions and (ii) that we capture all subframeworks with the new row.

To check the conditions for partial completeness, if $L'(a) = I$, then all attackers in subframeworks in $\mathcal{X}_{r'}$ are labeled "out", by assumption of correctness of the child node. If $L'(a) = O$, we need to make sure that there is an "in" attacker. This is the case for all subframeworks in $\mathcal{X}_{r'}$ if $lw(a) = O$. Analogously, we verify the partial completeness condition for $L'(a) = U$. If one of the checks fails, the row r' is discarded, which we denoted by $\sigma_{forget}(s, w, a)$. By $s-a$ and $w-a$ we denote the removal of a from structure (and incident attacks) and witness.

Finally, after forgetting an argument in rows r', multiple rows can coincide on their (s, w) part: these need to be summed (they "collapse" into a single row).

Example 8. *Let us consider node 13 which forgets a. Here, Algorithm 3 first filters out rows $r' \in \tau'$ for which the labeling in s' disagrees with lw' on a. A row r' is invalid for further consideration if it labeled a by O, while lw' has not witnessed a being attacked by an argument labeled I. Analogously for rows where a was labeled undecided.*

Join Nodes The idea of join nodes t is to "merge" tables from two children nodes t_1, t_2, with different $B_{\leq t_1}$ and $B_{\leq t_2}$. The basic principle is that if there is a row $r_1 = (s_1, w_1, p_1)$ and a row $r_2 = (s_2, w_2, p_2)$ from the two children, then if each row represents the sum of probabilities of $\mathcal{X}_{r_1}$ and $\mathcal{X}_{r_2}$, respectively, then we can merge these two whenever $s_1 = s_2$ (the subframeworks in $\mathcal{X}_{r_1}$ and $\mathcal{X}_{r_2}$ are compatible) and (i) compute $w = w_1 \cup w_2$ (we find witnesses for an argument being out or undecided from the respective subframeworks in at least one branch) and (ii) construct p by $\texttt{comp}(p_1, p_1, s)$, with s containing $F = (A, R, P)$. Define

$$common(s) = \prod_{a \in A} P(a) \cdot \prod_{a \in B_t \setminus A} (1 - P(a)) \cdot \prod_{r \in R} P(r) \cdot \prod_{(a,b) \in D \setminus R} (1 - P(r))$$

$$\texttt{comp}(p_1, p_2, s) = \frac{p_1 \cdot p_2}{common(s)}$$

Algorithm 3 Forget(τ', B, S)

1: $\tau'' := ((s - a, w - a, p) \in \tau' \mid \sigma_{forget}(s, w, a)$ is true)
2: **for** $(s, w) \in \{(s', w') \mid (s', w', p') \in \tau''\}$
3: $\quad p = \sum_{(s,w,p') \in \tau''} p'$
4: $\quad \tau := \tau \cup \{(s, w, p)\}$
5: **return** τ

Algorithm 4 Join(τ_1, τ_2, B)

1: $\tau := \{(s, w_1 \cup w_2, \texttt{comp}(p_1, p_2, s)) \mid (s, w_1, p_1) \in \tau_1, (s, w_2, p_2) \in \tau_2\}$
2: $\tau := \texttt{Merge}(\tau)$
3: **return** τ

with D being the set of all possible attacks incident to the current bag. That is, p is computed by the product of p_1 and p_2, but we have to discount the common part (otherwise the part of F that is the current bag would be counted twice). In case two rows with same s and w are created, `merge` merges these into one with summed probabilities.

6 Experimental Evaluation

We evaluate our algorithm for P-Ext, focusing on the $\mathsf{FP}^{\#\mathsf{P}}$-complete variant for complete semantics.

Implementations We implemented our algorithm in a Python (3.9.7) prototype.[1] As suggested by Dewoprabowo et al. (2022), we partially make use of database tools: the most expensive operations (join and forget) make use of a Python library Pandas (version 2.2.2) for database management.

Due to operations on small numbers, we implemented two variants, one using floating-point numbers dubbed `TD-Ext-f`, and one using rational numbers, called `TD-Ext-r`. For computing nice tree-decompositions, we used the `htd` library (Abseher, Musliu, and Woltran 2017).

To the best of our knowledge, there are no competing systems available that are tailored towards performance. For a baseline comparison, we implemented two more systems: (i) a naive brute-force algorithm enumerating pairs of subframeworks and extensions in Python and (ii) and answer set programming (ASP) (Gelfond and Lifschitz 1988; Niemelä 1999) based implementation `ASP-#Ext` for *counting* subframeworks where S is complete. Note that this implementation does not solve the same problem, but a simpler one, and is intended mainly as a basis for comparison.

Instances There is currently no standard benchmark set for PAFs. We constructed PAFs with a controlled tree-width and focused on instances not solved via preprocessing, i.e., do not contain many certain arguments and attacks. We constructed PAFs based on grids (k, n) with possible attacks (bidirectional, one-directional, or no attack) for each horizontal or vertical neighbour. The direction or absence of each attack was chosen with uniform probability. Argument and attack probabilities are picked from $\{0.1, 0.2, ..., 1\}$, with probabilities $\frac{10}{91}$ for uncertain and $\frac{1}{91}$ for 1. We let $k \in \{3, 4, 5, 6, 7\}$ and $n \in \{5, 10, 20, 50, 75, 100, 150\}$. Tree-width of such grid-graphs is bounded by $min(k, n)$. For each (k, n) we generated 4 PAFs and a set S to be checked by selecting $a \in A$ with probability 0.04. In total, there are 140 generated PAF instances, 28 for each k.

[1] Available at https://gitlab.tugraz.at/krr/paf-td.

k	TD-Ext-f	TD-Ext-r	ASP-#Ext
3	10.72 (0)	10.43 (0)	0.01 (21)
4	20.32 (0)	20.14 (0)	0.20 (22)
5	68.26 (0)	66.54 (0)	0.37 (23)
6	12.76 (9)	17.51 (8)	9.61 (25)
7	32.54 (10)	31.36 (8)	1.33 (24)

Table 1: Median running times in seconds (timeouts) for complete.

The number of arguments in a PAF instance is equal to the corresponding grid size. The minimum and maximum number of attacks created for instances of a given k were (23, 760) for $k = 3$, (26, 1045) for $k = 4$, (35, 1366) for $k = 5$, (44, 1686) for $k = 6$, and (50, 1979) for $k = 7$.

Setup and Results The experiments were carried out on a Linux machine (64-bit), with an 8-core i5 Intel CPU and 16 GB of memory. We enforced a runtime of maximum 300 seconds and a memory limit of 8192 MB per run.

In Table 1 we show the median running times and number of timeouts of solved instances. In the results, our approach scales with tree-width: no timeouts were reported with instances having a tree-width of at most 5. Note that PAFs with $k = 5$ include ones with $n = 150$ and $|A| = 750$. For a baseline comparison, ASP-#Ext showed significantly more timeouts. We speculate this is due to the high number of subframeworks, e.g., one instance reported more than $6 \cdot 10^7$ subframeworks. We opted not to include the naive (enumerating all subframeworks and extensions) approach in Table 1, as it did not terminate on the generated instances.

Regarding precision, using floats the results, in fact, differ to fractions: in four instances the former reported 0, while the result is higher than 0 (although low). Moreover, in one PAF TD-Ext-f reported $9.6 \cdot 10^{-15}$ and TD-Ext-r returned $7 \cdot 10^{-15}$. While the absolute difference is low, the percentage the former approach is off is roughly one-third. On the other hand, in many runs the results reported were close. We speculate that this is due to having many operations summing or multiplying in the dynamic programming algorithms. Due to similar performance of TD-Ext-f and TD-Ext-r, it does appear to give TD-Ext-r an edge.

7 Extensions

Our algorithm presented in Section 5 can be extended, and we discuss here possible extensions.

First, in addition to enforcing arguments to be "in" (as in the P-Ext problem), we can also directly enforce arguments to be O or U.

This is direct in the algorithm: in introduction nodes we can restrict generation of rows to exactly those fitting the chosen criteria. For instance, only constructing labelings with a chosen argument assigned "out" or "undecided".

Second, as discussed by Fazzinga, Flesca, and Furfaro (2019), we can introduce dependencies between arguments. For instance, consider the case that whenever a subframework contains an argument a then such a subframework also must contain an argument b. Then, as discussed by Fazzinga, Flesca, and Furfaro, one can specify marginal probabilities of the three different cases: containing neither argument, containing only b and containing both.

This can be achieved, by considering, in the tree-decomposition generation, an additional edge between a and b, representing an additional dependency. This additional edge ensures that a or b can only be forgotten if the other argument was already seen.

Then computation of probabilities must be deferred until both arguments a and b are in a bag (only then can we compute the marginal probability of this case). As discussed earlier (Wallner 2020), implication dependencies such as these can be important for representing internal structure of arguments, while at the same time, as discussed by Fazzinga, Flesca, and Furfaro, allowing for compact representation of probabilities by marginal probabilities. For instance, if an argument a is a sub-argument of b, when considering the internal structure of these arguments, then having b but not a might be unwarranted. Or, for instance, if the contents of arguments a and b together imply existence of an argument c, then dependency edges between all can be added, and marginal probabilities of the different cases considered.

Finally, to adapt Algorithm 1 to admissible sets or stable extensions, the condition for semantics can be directly adapted: for admissibility the checks for undecidedness can be omitted, while for stable semantics, introduction nodes are not allowed to assign arguments to undecided. We remark that computing the probability of a set of arguments being admissible or stable is not $\mathsf{FP}^{\#\mathsf{P}}$-hard (Fazzinga, Flesca, and Parisi 2015).

8 Discussion

In this work we revisited computationally complex problems in probabilistic argumentation under the constellation approach. We refined earlier complexity results, showing a divergence between complexities of two main reasoning tasks under their counting variants. For the problem of computing the probability of a set of arguments being complete, we developed a dynamic programming algorithm. Our experimental evaluation shows promise of the approach, e.g., PAFs with up to 750 arguments were solved by our prototype, depending on the attack-structure and tree-width.

Algorithmic approaches utilizing tree-decompositions were investigated before for formal argumentation (Dunne 2007; Dvořák et al. 2022; Dvořák et al. 2011; Dvořák, Pichler, and Woltran 2012; Dvořák, Szeider, and Woltran 2012; Fichte et al. 2021; Lampis, Mengel, and Mitsou 2018; Popescu and Wallner 2023), also for counting extensions containing queried arguments in AFs (Fichte, Hecher, and Meier 2019), and, e.g., for model counting in Boolean logic (Samer and Szeider 2010) and weighted model counting (Fichte et al. 2018). In contrast, we present a novel dynamic programming algorithm for probabilistic argumentation under the constellation approach.

For future work, an interesting avenue is to extend our work for acceptability of arguments in PAFs, under the constellation approach. We think that our Algorithm 1 provides a useful basis for computing acceptability. The main barrier is to avoid over-computation, or over-summation. For in-

stance, by slightly relaxing our algorithm to have a queried argument to be assigned "in", and allowing also other arguments to be "in" we can extend the algorithm to consider also other extensions that contain a queried argument. However, directly counting all pairs of subframeworks and labelings assigning a queried argument to be "in" over-counts, since there might be many such labelings in a particular subframework.

We think that this way of over-counting can be addressed in a dynamic programming way operating on tree-decompositions, but requires more work for a detailed investigation. We remark that Theorem 5 (hardness of acyclic PAFs) does not prevent existence of algorithms for acceptability using tree-decompositions: acyclic PAFs may have a tree-width higher than one. Nevertheless, our complexity results suggest that acceptability is more challenging.

In addition, computing the probability of a set of arguments being a preferred extension, i.e., a subset maximal complete extension, faces a different challenge: here we need to ensure that only subset maximal complete (admissible) extensions are considered. Subset maximality has been considered for dynamic programming algorithms operating on tree-decompositions, e.g., in the D-FLAT framework (Bliem et al. 2016). We think it is an interesting direction to incorporate such findings for tackling the preferred semantics.

Similarly, we think that considering further variants of acceptability of an argument, e.g., including skeptical acceptance (an argument must be in all extensions of a chosen semantics) and allowing for probabilities to be attached to extensions (Alfano et al. 2023), presents intriguing challenges for dynamic programming approaches.

As a more general direction, we think that future work, to advance complex reasoning in probabilistic argumentation further, should include also different forms of optimizations. For instance, an (NP-hard) preprocessing might be beneficial for dynamic programming algorithms that can restrict the number of rows in tables, e.g, by pre-computing impossible combinations (e.g., argument a is never "in" in any subframework). While these problems can be complex themselves, we think that they can overall be used to optimize tree-decomposition-based algorithms.

Acknowledgements

This research was funded in whole or in part by the Austrian Science Fund (FWF) by grant P35632.

References

Abseher, M.; Musliu, N.; and Woltran, S. 2017. htd - A free, open-source framework for (customized) tree decompositions and beyond. In Salvagnin, D., and Lombardi, M., eds., *Proc. CPAIOR*, volume 10335 of *Lecture Notes in Computer Science*, 376–386. Springer.

Alfano, G.; Calautti, M.; Greco, S.; Parisi, F.; and Trubitsyna, I. 2023. Explainable acceptance in probabilistic and incomplete abstract argumentation frameworks. *Artif. Intell.* 323:103967.

Amgoud, L.; Cayrol, C.; Lagasquie-Schiex, M.; and Livet, P. 2008. On bipolarity in argumentation frameworks. *Int. J. Intell. Syst.* 23(10):1062–1093.

Atkinson, K.; Baroni, P.; Giacomin, M.; Hunter, A.; Prakken, H.; Reed, C.; Simari, G. R.; Thimm, M.; and Villata, S. 2017. Towards artificial argumentation. *AI Mag.* 38(3):25–36.

Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds. 2018. *Handbook of Formal Argumentation*. College Publications.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowl. Eng. Rev.* 26(4):365–410.

Baumeister, D.; Neugebauer, D.; Rothe, J.; and Schadrack, H. 2018. Verification in incomplete argumentation frameworks. *Artif. Intell.* 264:1–26.

Besnard, P.; García, A. J.; Hunter, A.; Modgil, S.; Prakken, H.; Simari, G. R.; and Toni, F. 2014. Introduction to structured argumentation. *Argument Comput.* 5(1):1–4.

Biere, A.; Järvisalo, M.; and Kiesl, B. 2021. Preprocessing in SAT solving. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 391–435.

Bliem, B.; Charwat, G.; Hecher, M.; and Woltran, S. 2016. D-flat^2: Subset minimization in dynamic programming on tree decompositions made easy. *Fundam. Informaticae* 147(1):27–61.

Bodlaender, H. L., and Koster, A. M. C. A. 2008. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.* 51(3):255–269.

Bodlaender, H. L. 1993. A tourist guide through treewidth. *Acta Cybern.* 11(1-2):1–21.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Brewka, G.; Polberg, S.; and Woltran, S. 2014. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intell. Syst.* 29(1):30–38.

Caminada, M. W. A., and Gabbay, D. M. 2009. A logical account of formal argumentation. *Stud Logica* 93(2-3):109–145.

Cyras, K.; Oliveira, T.; Karamlou, A.; and Toni, F. 2021. Assumption-based argumentation with preferences and goals for patient-centric reasoning with interacting clinical guidelines. *Argument Comput.* 12(2):149–189.

Dewoprabowo, R.; Fichte, J. K.; Gorczyca, P. J.; and Hecher, M. 2022. A practical account into counting dung's extensions by dynamic programming. In Gottlob, G.; Inclezan, D.; and Maratea, M., eds., *Proc. LPNMR*, volume 13416 of *Lecture Notes in Computer Science*, 387–400. Springer.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–358.

Dunne, P. E.; Hunter, A.; McBurney, P.; Parsons, S.; and Wooldridge, M. J. 2011. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artif. Intell.* 175(2):457–486.

Dunne, P. E. 2007. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.* 171(10-15):701–729.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *Theor. Comput. Sci.* 340(3):496–513.

Dvořák, W., and Dunne, P. E. 2018. Computational problems in formal argumentation and their complexity. In Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds., *Handbook of Formal Argumentation*. College Publications. chapter 13, 631–688.

Dvořák, W.; Morak, M.; Nopp, C.; and Woltran, S. 2011. dynPARTIX - A dynamic programming reasoner for abstract argumentation. In Tompits, H.; Abreu, S.; Oetsch, J.; Pührer, J.; Seipel, D.; Umeda, M.; and Wolf, A., eds., *Proc. INAP, Revised Selected Papers*, volume 7773 of *Lecture Notes in Computer Science*, 259–268. Springer.

Dvořák, W.; Hecher, M.; König, M.; Schidler, A.; Szeider, S.; and Woltran, S. 2022. Tractable abstract argumentation via backdoor-treewidth. In *Proc. AAAI*, 5608–5615. AAAI Press.

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.* 186:1–37.

Dvořák, W.; Szeider, S.; and Woltran, S. 2012. Abstract argumentation via monadic second order logic. In Hüllermeier, E.; Link, S.; Fober, T.; and Seeger, B., eds., *Proc. SUM*, volume 7520 of *Lecture Notes in Computer Science*, 85–98. Springer.

Fan, X.; Toni, F.; Mocanu, A.; and Williams, M. 2014. Dialogical two-agent decision making with assumption-based argumentation. In Bazzan, A. L. C.; Huhns, M. N.; Lomuscio, A.; and Scerri, P., eds., *Proc. AAMAS*, 533–540. IFAAMAS/ACM.

Fazzinga, B.; Flesca, S.; Furfaro, F.; and Scala, F. 2019. Efficiently computing extensions' probabilities over probabilistic bipolar abstract argumentation frameworks. *Intelligenza Artificiale* 13(2):189–200.

Fazzinga, B.; Flesca, S.; Furfaro, F.; and Monterosso, G. 2024. Quantitative reasoning over incomplete abstract argumentation frameworks. In Larson, K., ed., *Proc. IJCAI*, 3360–3368. ijcai.org.

Fazzinga, B.; Flesca, S.; and Furfaro, F. 2018. Credulous and skeptical acceptability in probabilistic abstract argumentation: complexity results. *Intelligenza Artificiale* 12(2):181–191.

Fazzinga, B.; Flesca, S.; and Furfaro, F. 2019. Complexity of fundamental problems in probabilistic abstract argumentation: Beyond independence. *Artif. Intell.* 268:1–29.

Fazzinga, B.; Flesca, S.; and Furfaro, F. 2020. Revisiting the notion of extension over incomplete abstract argumentation frameworks. In Bessiere, C., ed., *Proc. IJCAI*, 1712–1718. ijcai.org.

Fazzinga, B.; Flesca, S.; and Parisi, F. 2015. On the complexity of probabilistic abstract argumentation frameworks. *ACM Trans. Comput. Log.* 16(3):22:1–22:39.

Fichte, J. K.; Hecher, M.; Woltran, S.; and Zisser, M. 2018. Weighted model counting on the GPU by exploiting small treewidth. In Azar, Y.; Bast, H.; and Herman, G., eds., *Proc. ESA*, volume 112 of *LIPIcs*, 28:1–28:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Fichte, J. K.; Hecher, M.; Mahmood, Y.; and Meier, A. 2021. Decomposition-guided reductions for argumentation and treewidth. In Zhou, Z., ed., *Proc. IJCAI*, 1880–1886. ijcai.org.

Fichte, J. K.; Hecher, M.; and Meier, A. 2019. Counting complexity for reasoning in abstract argumentation. In *Proc. AAAI*, 2827–2834. AAAI Press.

Fox, J.; Glasspool, D.; Grecu, D.; Modgil, S.; South, M.; and Patkar, V. 2007. Argumentation-based inference and decision making–a medical perspective. *IEEE Intell. Syst.* 22(6):34–41.

Gabbay, D.; Giacomin, M.; Simari, G. R.; and Thimm, M., eds. 2021. *Handbook of Formal Argumentation*, volume 2. College Publications.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. ICLP/SLP*, 1070–1080. MIT Press.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2021. Model counting. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 993–1014.

Hemaspaandra, L. A., and Vollmer, H. 1995. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News* 26(1):2–13.

Hunter, A., and Thimm, M. 2017. Probabilistic reasoning with abstract argumentation frameworks. *J. Artif. Intell. Res.* 59:565–611.

Hunter, A.; Polberg, S.; Potyka, N.; Rienstra, T.; and Thimm, M. 2021. Probabilistic argumentation: a survey. In Gabbay, D.; Giacomin, M.; Simari, G. R.; and Thimm, M., eds., *Handbook of Formal Argumentation*, volume 2. College Publications. chapter 7, 397–441.

Hunter, A. 2013. A probabilistic approach to modelling uncertain logical arguments. *Int. J. Approx. Reason.* 54(1):47–81.

Kloks, T. 1994. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer.

Lampis, M.; Mengel, S.; and Mitsou, V. 2018. QBF as an alternative to Courcelle's theorem. In Beyersdorff, O., and Wintersteiger, C. M., eds., *Proc. SAT*, volume 10929 of *Lecture Notes in Computer Science*, 235–252. Springer.

Li, H.; Oren, N.; and Norman, T. J. 2011. Probabilistic argumentation frameworks. In Modgil, S.; Oren, N.; and Toni, F., eds., *Proc. TAFA*, volume 7132 of *Lecture Notes in Computer Science*, 1–16. Springer.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.* 25(3-4):241–273.

Papadimitriou, C. H. 2007. *Computational complexity*. Academic Internet Publ.

Popescu, A., and Wallner, J. P. 2023. Reasoning in assumption-based argumentation using tree-decompositions. In Gaggl, S. A.; Martinez, M. V.; and Ortiz, M., eds., *Proc. JELIA*, volume 14281 of *Lecture Notes in Computer Science*, 192–208. Springer.

Prakken, H., and Sartor, G. 2015. Law and logic: A review from an argumentation perspective. *Artif. Intell.* 227:214–245.

Samer, M., and Szeider, S. 2010. Algorithms for propositional model counting. *J. Discrete Algorithms* 8(1):50–64.

Valiant, L. G. 1979. The complexity of computing the permanent. *Theor. Comput. Sci.* 8:189–201.

Wallner, J. P. 2020. Structural constraints for dynamic operators in abstract argumentation. *Argument Comput.* 11(1-2):151–190.