UNIVERSITÀ DELLA CALABRIA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

THESIS SUMMARY

Analysis and implementation of hard cases in Probabilistic Argumentation Frameworks

RELATORI

Prof. Sergio FLESCA

CANDIDATO

Luca BRUNO

Matr. 145994

Anno Accademico 2012-2013

Contents

1 Introduction

1.1 Goals of this work

The purpose of this work is to study argumentation frameworks and its variants[8, 11, 12], in particular probabilistic argumentation frameworks ($PrAF$). Then we analyze an existing approach based on Monte Carlo simulation to compute the probability that a set of arguments is an extension for a given PrAF according to some semantics.

It will be shown that for some properties this probability can be computed in polynomial time.

Based on these tractable cases, we will extend the existing Monte Carlo approach to optimize harder cases. We will show that the new algorithm is generally more efficient compared to the naive algorithm, and especially efficient for the complete semantics.

1.2 Why argumentation frameworks

The purpose of argumentation is to search for the requirements that make an argument correct by examining the errors of reasoning we make when we try to use arguments[5]. The task of evaluation is to determine whether an argument is weak or strong by general criteria that can be applied to it.

There is no strong agreement in the literature of argumentation theory on how to define an argument. Some definitions are more minimal while others are more inclusive. A minimal definition is the following: an *argument* is a set of statements (propositions), made up of a conclusion, a set of premises, and an inference from the premises to the conclusion. An argument can be supported by other arguments, or it can be attacked by other arguments.

The general approach or methodology of argumentation is different than the traditional deductive approach. The deductive logic concentrated on a single inference, where the premises and conclusion are designated in advance, and applied formal models like propositional calculus and quantification theory to determine whether the conclusion follows from the premises. In contrast, the argumentation approach

looks at two sides of an argument, the pro and the contra, to examine how arguments interact with each other. By this process of examining one argument against the other, the weaknesses in each argument are revealed, and it is shown which of the two arguments is the stronger.

2 Argumentation frameworks

2.1 Abstract argumentation frameworks

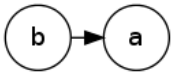
Dung's abstract argumentation frameworks (AAFs) [2] are widely used in formal approaches to argumentation [13, 6]. They provide several standard semantics, each capturing different intuitions about how to handle conflicts among arguments. This makes them a highly useful tool in argumentation and algorithms for computing extensions that have received considerable interest.

An abstract argument system or argumentation framework, as introduced in a seminal paper by Dung, is simply a pair $\langle Arg, Def \rangle$ where Arg is a set of abstract arguments and Def describes the attack relations between arguments. The set Arg may be finite or infinite in general, however in this work we will concentrate on finite sets of arguments. An argumentation framework has an obvious representation as a directed graph where nodes are arguments and edges are drawn from attacking to attacked arguments.

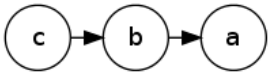
While the word *argument* may recall several different meanings, abstract argument systems are not bound to any of them: an *abstract argument* is anything that may attack or be attacked by another argument. The attack relation is also known as *defeat* relation.

Given a set of arguments and attack relations, we want to decide the *justification state* of an argument or a set of arguments. Intuitively, an argument is justified if it can survive to attacks of other arguments.

A simple example of argumentation framework $\langle \{a, b\}, \{\langle b, a \rangle\} \rangle$ is shown in figure 2.1a. In this case argument b is justified because it is not attacked by any other argument, while argument a is not justified because it is attacked by argument b.



(a) Argument b attacks argument a



(b) Argument a is defended by argument c

In figure 2.1b, argument a is justified even though it is attacked by b, because it is defended by argument c. In fact, since b is not justified, its attack has no validity over argument a.

Definition 1. A Dung's abstract argumentation framework (AAF in short) is a pair $\langle Arg, Def \rangle$ where Arg is a set of arguments and $Def \subseteq Arg \times Arg$ is a set of defeats between arguments.

One of the fundamental properties that a set of arguments $S \subseteq Arg$ must satisfy to be justified is the notion of being *conflict-free*. A set must contain only arguments that do not conflict with each other, in the sense that an argument $a \in S$ must not attack any other argument $b \in S$.

Definition 2. Given an AAF $\langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is said to be *conflict-free* iff $\nexists a, b \in S$ such that $\langle a, b \rangle \in Def$.

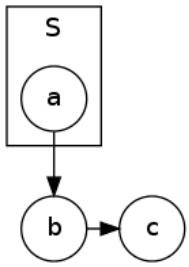
The second fundamental property is that a set of arguments $S \subseteq Arg$ must be *admissible*. Before formalizing this concept, we first define the acceptability[4] of an argument with respect to a set S.

Definition 3. Given an AAF $\langle Arg, Def \rangle$ and a set of arguments $S \subseteq Arg$, an argument $a \in Arg$ is *acceptable with respect to* S iff $\forall b \in Arg$ such that $\langle b, a \rangle \in Def$ then $\exists c \in S$ such that $\langle c, b \rangle \in Def$.

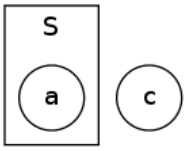
In other words, an argument $a \in A$ is acceptable with respect to a set of arguments $S \subseteq Arg$ if it is defended by S. Some cases of argument acceptability are shown in figure 2.1.

Definition 4. Given an AAF $\langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is *admissible* iff S is conflict-free and $\forall a \in S$, a is acceptable with respect to S.

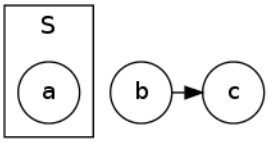
Intuitively, a set of arguments S is admissible if it is able to defend itself from the attacks of any other arguments. Some cases of admissibility are shown in figure 2.2.



(a) Argument c is acceptable w.r.t. S

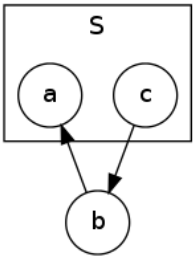


(b) Argument c is acceptable w.r.t. S

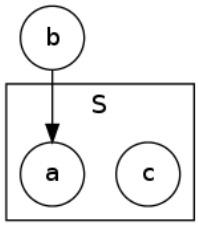


(c) Argument c is not acceptable w.r.t. S

Figure 2.1: Accettabilità argomenti rispetto ad S



(a) S is admissible



(b) S is not admissible

Figure 2.2: Admissibility of S

2.2 Semantics of abstract argumentation frameworks

An argumentation semantics is the formal definition of a method (either declarative or procedural) ruling the argument evaluation process. We focus on the extension-based approach, where a semantics definition specifies how to derive a set of extensions from an argumentation framework, where an extension E of an argumentation framework $\mathcal{F} = \langle Arg, Def \rangle$ is simply a subset of Arg, intuitively representing a set of arguments which can "survive together", are "collectively acceptable" or in general that are justified with regards to some criteria.

Different semantics have been developed that lead to different notions of exten-

sions. First we define the most general concept of semantic evaluation function and extensions.

Definition 5. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$ and a semantics $\mathfrak{S}$, a *semantic evaluation function* $\xi_{\mathfrak{S}}(\mathcal{F}, E)$ returns true iff the set of arguments $E \subseteq Arg$ is deemed consistent using the semantics $\mathfrak{S}$ when evaluated over the argument system $\mathcal{F}$, that is E is an extension for $\mathcal{F}$ under the semantics $\mathfrak{S}$. The set of all extensions prescribed by semantics $\mathfrak{S}$ for $\mathcal{F}$ is denoted as $\mathcal{E}_{\mathfrak{S}}(\mathcal{F}) \subseteq 2^{Arg}$.

Also we formalize the notion of *justification state* for a single argument in an argumentation framework with respect to a given semantics.

Definition 6. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, an argument $a \in Arg$ is said to be *justified* under the semantics $\mathfrak{S}$ iff $\forall E \in \mathcal{E}_{\mathfrak{S}}(\mathcal{F})$, $a \in E$.

Given the basic definitions of admissibility in the previous section and of extension above, in this work we will focus on traditional types of extension defined below, namely: stable, preferred, complete, grounded and ideal.

Definition 7. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is a *complete* extension iff S is admissible and $\forall a \in Arg$ such that a is acceptable with respect to S, $a \in S$.

Or in other words, S contains all the arguments that are acceptable with respect to S, that is $\nexists a \in Arg \setminus S$ such that a is acceptable with respect to S. It is easy to see that deciding whether a set of arguments is complete can be computed in polynomial time with respect to the number of arguments and defeats.

Definition 8. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is the *grounded* extension iff S is a minimal (w.r.t. $\subseteq$) complete set of arguments, it is unique and is denoted as $GE(\mathcal{F})$.

Deciding whether a set of arguments is grounded is certainly harder than deciding if a set is complete, and in particular it requires exponential time with respect to the number of arguments and defeats.

A stronger requirement than complete semantics is the stable semantics.

Definition 9. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is a *stable* extension iff S is conflict-free and $\forall a \in Arg \setminus S$, $\exists b \in S$ such that $\langle b, a \rangle \in Def$.

A stable extension $E \subseteq Arg$ must defeat all arguments that are not in the set. By definition, any stable extension is also a complete extension and a maximal conflict-free set of $\mathcal{F}$.

Definition 10. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is a *preferred* extension iff S is a maximal (w.r.t. $\subseteq$) admissible set of arguments.

Preferred semantics relax the requirement of stable semantics to attack everything outside of the set. A preferred set is a maximal admissible set, in the sense that it's a largest set able to defend itself from attacks.

Definition 11. Given an AAF $\mathcal{F} = \langle Arg, Def \rangle$, a set of arguments $S \subseteq Arg$ is an *ideal extension* iff S is admissible and $\forall E \in \mathcal{E}_{preferred}(\mathcal{F})$, $S \subseteq E$.

The grounded set is an ideal set. Ideal sets that are strictly larger than the grounded set may exist.

The relationships between the above semantics[12] is shown in figure 2.3.

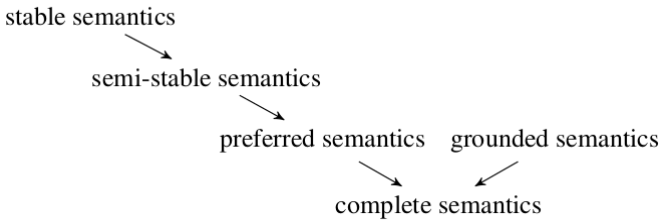


Figure 2.3: Relationships between semantics

2.3 Probabilistic argumentation frameworks

Likelihoods and probabilities are used to introduce uncertainty to reasoning in complex domains. In argumentation frameworks, the uncertainty affects the interactions between arguments and the strength of conclusions. Within argument systems, an

argumentation semantics defines a method by which a set of justified arguments can be deduced. As a reasoning approach, a semantics takes an argumentation framework as its knowledge base and produces a set of justified arguments as its output. The problem addressed by probabilistic argumentation frameworks is to introduce such uncertainty in argumentation frameworks and to identify the effects of probabilities on argument justification.

The approach is very simple. Dung's abstract argumentation frameworks are extended by introducing probabilities assigned to arguments and defeats [8]. The semantics of such a framework identifies the likelihood of different sets of arguments being justified according to different types of extensions.

A probabilistic argumentation framework ($PrAF$ for short) represents an entire set of AAFs that can potentially exist. A specific AAF has a certain likelihood of being *induced* from the PrAF.

Definition 12. A probabilistic argumentation framework is a tuple $\langle A, P_A, D, P_D \rangle$ where $\langle A, D \rangle$ is an AAF, $P_A : A \to (0 : 1]$ and $P_D : D \to (0 : 1]$.

The lower bound of these probabilities is not 0 because any argument or defeat with a likelihood of 0 cannot ever appear within a AAF induced from the PrAF.

The process of creating an AAFs that can be induced from the PrAF is called the *inducement* of a AAF from the PrAF. All arguments and defeats with a likelihood of 1 will be found in the induced AAF, which can then contain additional arguments and defeats, as specified by the following definition.

Definition 13. An AAF $\langle Arg, Def \rangle$ (or possible world) is said to be *induced* from a PrAF $\langle A, P_A, D, P_D \rangle$ iff all of the following hold:

- $Arg \subseteq A$
- $Def \subseteq D \cap (Arg \times Arg)$
- $\forall a \in A$ such that $P_A(a) = 1$, $a \in Arg$
- $\forall \langle f, t \rangle \in D$ such that $P_D(f, t) = 1$, $\langle f, t \rangle \in Def$

The goal is to compute the likelihood that some set of arguments exists and is justified according to some semantics within the AAFs induced from a PrAF. A critical simplifying assumption is made, namely that the likelihood of one argument or defeat appearing in an induced AAF is independent of the likelihood of some other argument or defeat appearing in the same AAF.

The P_D relation associates a conditional probability with each possible defeat over the existance of the incident arguments. That is, for some arguments a, b and a generic induced AAF $\langle Arg, Def \rangle$:

$$P_D(a,b) = P(\langle a,b \rangle \in Def \mid a,b \in Arg)$$

The probability $P_{PrAF}(\mathcal{F})$ of some $\mathcal{F} = \langle Arg, Def \rangle$ being induced from a $PrAF = \langle A, P_A, D, P_D \rangle$ can be computed via the joint probabilities of the arguments and defeat relations as follows:

$$P_{PrAF}(\mathcal{F}) = \prod_{a \in Arg} P_A(a) \cdot \prod_{a \in Arg \setminus A} (1 - P_A(a)) \cdot \prod_{d \in Def} P_D(d) \cdot \prod_{d \in DefA \setminus Def} (1 - P_D(d))$$

where $DefA = \{\langle a,b \rangle \mid a,b \in Arg \wedge \langle a,b \rangle \in D\}$ is the set of all possible defeats that can appear in an induced AAF.

The set of all possible AAFs that can be induced from a PrAF form a partitioned event space for which $\sum_{AAF} P_{\mathcal{F}}(AAF) = 1$ holds.

Given a $PrAF = \langle A, P_A, D, P_D \rangle$ and a semantics $\mathfrak{S}$, the likelihood of a set of arguments S being justified according to $\mathfrak{S}$ is defined as follows:

$$P_{\mathfrak{S}}(S) = \sum_{\mathcal{F}} P_{PrAF}(\mathcal{F}) \; where \; \xi_{\mathfrak{S}}(\mathcal{F}, S) = true$$

The size of the set of possible AAFs which can be induced from a PrAF grows exponentially with regards to the number of arguments and defeats within the PrAF.

3 Computing the probability that a set of arguments is an extension

3.1 Monte Carlo simulation

In this section we describe a Monte Carlo simulation based approach to compute $P_{\mathfrak{S}}(S)$ for an arbitrary $PrAF$, where $\mathfrak{S}$ is a generic semantics for which a semantic evaluation function $\mathcal{E}_{\mathfrak{S}}$ exists. A Monte Carlo simulation operates by repeatedly sampling a distribution many times in order to approximate it. More specifically, such a simulation has three basic steps. First, given a possible set of inputs, a subset of these inputs is selected according to some probability distribution. Second, some computation is performed using the selected inputs. Finally, the results of repeating the first two steps multiple times is aggregated. Monte Carlo simulation has a long history, and has been applied to a variety of computationally difficult problems including inference in Bayesian Networks[9], reinforcement learning[10] and game playing[7]. In this context of probabilistic argumentation frameworks, this process involves randomly inducing AAFs from a PrAF by sampling the space of possible AAFs in a way that approximates the AAFs true distribution in the probability space.

The algorithm 1 samples N AAFs from the set of inducible AAFs. A single AAF is generated by randomly selecting arguments and defeats according to their likelihood of appearance. This resultant AAF is then evaluated for the presence of S as extension through the use of the semantic evaluation function $\xi_{\mathfrak{S}}$, and if this function holds, the AAF is counted. The probability $P_{\mathfrak{S}}(S)$ is finally approximated as the ratio of the total number of AAFs in which $\xi_{\mathfrak{S}}$ holds to the number AAFs sampled. As the number of trials increases, the error in the approximation of $P_{\mathfrak{S}}(S)$ shrinks.

Given X the number of observed successes, N the number of trials and $p = X/N$ the observed mean, the confidence interval for the real $P_{\mathfrak{S}}(S)$ is defined as follows[3]:

$$P_{\mathfrak{S}}(S) = p \pm z_{1-\alpha/2} \cdot \sqrt{\frac{p(1-p)}{N}} \tag{3.1}$$

Algorithm 1 Algorithm for approximating $P_{PrAF}(X)$

Require: A Probabilistic Argumentation Framework $PrAF = (A, PA, D, PD)$
Require: A set of arguments $S \subseteq A$
Require: A number of trials $N \in \mathbb{N}$
Require: A semantic evaluation function ξ
$X = 0$
$\forall I = 0..N$ **do**
    $Arg = Def = \emptyset$
    $\forall a \in A$ **do**
        Generate a random number $r \in [0, 1]$
        if $r \leq P_A(a)$ **then**
            $Arg = Arg \cup \{a\}$
        end if
    end for
    $\forall \langle a, b \rangle \in D$ such that $a, b \in Arg$ **do**
        Generate a random number $r \in [0, 1]$
        if $r \leq P_D(\langle a, b \rangle)$ **then**
            $Def = Def \cup \{\langle a, b \rangle\}$
        end if
    end for
    if $\xi_{\mathfrak{S}}(\langle Arg, Def \rangle, S) = true$ **then**
        $X = X + 1$
    end if
end for
return $Count/N$

from which it is possible to derive the estimated number of trials for the given error level:

$$\epsilon = z_{1-\alpha/2} \cdot \sqrt{\frac{p(1-p)}{N}}$$

$$N = \frac{z^2_{1-\alpha/2} \cdot p \cdot (1-p)}{\epsilon^2} \tag{3.2}$$

The value of p after a single trial will be either 0 or 1, making the approximation problematic. To overcome this problem, the Agresti-Coull interval[1] is used. The

general form of this interval is the same as that of equation (3.1), except that p and N are substituted by p' and N' being computed as follows:

$$N' = N + z^2_{1-\alpha/2} \qquad p' = \frac{X + \left(z^2_{1-\alpha/2}/2\right)}{N'} \tag{3.3}$$

from which we can derive the new estimated number of trials to be used in algorithm 2:

$$N = \frac{z^2_{1-\alpha/2} \cdot p' \cdot (1-p')}{\epsilon^2} - z^2_{1-\alpha/2}$$

Algorithm 2 Algorithm for approximating $P_{PrAF}(X)$ given a confidence interval

Require: A Probabilistic Argumentation Framework $PrAF = (A, P_A, D, P_D)$
Require: A set of arguments $S \subseteq A$
Require: A confidence level $z_{1-\alpha/2}$
Require: An error level ϵ
Require: A semantic evaluation function ξ
$X = N = 0$
do
  Generate $\langle Arg, Def \rangle$ as in algorithm 1
  if $\xi_{\mathfrak{S}}(\langle Arg, Def \rangle, S) = true$ **then**
    $X = X + 1$
  end if
  $N = N + 1$
  $p' = \frac{X + \left(z^2_{1-\alpha/2}/2\right)}{N + z^2_{1-\alpha/2}}$
  $\hat{N} = \frac{z^2_{1-\alpha/2} \cdot p' \cdot \left(1-p'\right)}{\epsilon^2} - z^2_{1-\alpha/2}$
while $N \leq \hat{N}$
return X/N

3.2 Tractable cases

The number of AAFs that can be induced from a PrAF is certainly exponential in the number of arguments and defeats, but there are certain cases where computing the probability for a property does not require evaluating the whole space of induced

AAFs. In particular, it can be shown that the probability of a set of arguments being conflict-free and admissible can be computed in polynomial time with respect to the number of arguments and defeats.

Lemma 14. *Given* $PrAF = \langle A, P_A, D, P_D \rangle$ *and a set of arguments* $S \subseteq Arg$*, then* $P_{CF}(S) = P_1(S) \cdot P_2(S)$*, where:*

$$P_1(S) = \prod_{a \in S} P_A(a),$$

$$P_2(S) = \prod_{\substack{\langle b,c \rangle \in D \\ \wedge b,c \in S}} (1 - P_D(\langle b, c \rangle))$$

Lemma 14 derives from the fact that $CF(S)$ may expressed as the probabilistic event $E_{CF} = e_1 \wedge e_2$, where:

- e_1 is the event that all arguments in S occur;
- e_2 is the event that S is conflict-free.

Lemma 15. *Given* $PrAF = \langle A, P_A, D, P_D \rangle$ *and a set of arguments* $S \subseteq Arg$*, then* $P_{admissible}(S) = P_{CF}(S) \cdot P_3(S)$*, where:*

$$P_3(S) = \prod_{a \in A \setminus S} (P_{31}(S, a) + P_{32}(S, a) + P_{33}(S, a)),$$

$$P_{31}(S, a) = 1 - P_A(a),$$

$$P_{32}(S, a) = P_A(d) \cdot \prod_{\substack{\langle a,b \rangle \in D \\ \wedge b \in S}} (1 - P_D(\langle a, b \rangle)),$$

$$P_{33}(S, a) = P_A(d) \cdot \left(1 - \prod_{\substack{\langle a,b \rangle \in D \\ \wedge b \in S}} (1 - P_D(\langle a, b \rangle))\right) \cdot \left(1 - \prod_{\substack{\langle c,a \rangle \in D \\ \wedge c \in S}} (1 - P_D(\langle c, a \rangle))\right)$$

Lemma 15 extends lemma 14 to the case of admissibility. We introduce the joint event that all arguments in S are acceptable with respect to S; that is for each argument $a \notin A \setminus S$ one of the following independent events must hold:

- The argument a does not occur;
- The argument a occurs but does not defeat any argument in S;

- The argument a occurs and there exists at least one defeat from a to an argument $b \in S$, but there exists at least one defeat from an argument $c \in S$ to a, in which case S is able to defend itself.

3.3 Optimization of hard cases based on tractable cases

The fact that $P_{CF}(S)$ is computable in polynomial time with respect to the number of arguments and defeats, makes the computation of $P_{\mathfrak{S}}(S)$ more efficient for semantics that require the conflict-free principle.

We observe that $P_{\mathfrak{S}\cap CF}(S) = P_{\mathfrak{S}|CF}(S) \cdot P_{CF}(S)$, and that as long as $\mathcal{E}_{\mathfrak{S}}(\mathcal{F}) \subseteq \mathcal{E}_{CF}(\mathcal{F})$ then $P_{\mathfrak{S}}(S) = P_{\mathfrak{S}\cap CF}(S)$ holds. This is valid for all traditional types of extension, as the set of arguments is required to be at least conflict-free. Given the exact value of $P_{CF}(S)$ can be computed in polynomial time, we focus on the problem of estimating $P_{\mathfrak{S}|CF}(S)$ with Monte Carlo rather than $P_{\mathfrak{S}}(S)$.

Given X the number of observed successes, N the number of trials and $p = X/N$ the observed mean, the confidence interval for the real $P_{\mathfrak{S}}(S)$ is defined as follows:

$$P_{CF}(S) \cdot P_{\mathfrak{S}|CF}(S) = P_{CF}(S) \cdot \left(p \pm z_{1-\alpha/2} \cdot \sqrt{\frac{p(1-p)}{N}} \right)$$

from which we derive:

$$\epsilon = P_{CF}(S) \cdot z_{1-\alpha/2} \cdot \sqrt{\frac{p(1-p)}{N}}$$

$$N = \frac{z^2_{1-\alpha/2} \cdot p(1-p)}{\epsilon^2} \cdot P^2_{CF}(S)$$

Again, we use the Agresti-Coull interval, similar to equation (3.3), to obtain the following estimated number of trials:

$$N = \frac{z^2_{1-\alpha/2} \cdot p' \cdot (1-p')}{\epsilon^2} \cdot P^2_{CF}(S) - z^2_{1-\alpha/2}$$

This new formula will replace the old estimation in algorithm 2, together with

the returned value $P_{CF}(S) \cdot X/N$.

Theoretical efficiency of the new algorithm. Said $\hat{N}$ the number of trials and $\hat{p}$ the observed mean in the naive approach, we want to determine the new number of trials in terms of the naive trials.

For simplicity, we recall equation (3.2) without the Agresti-Coull perturbations:

$$\hat{N} = \frac{z^2_{1-\alpha/2} \cdot \hat{p} \cdot (1-\hat{p})}{\epsilon^2}$$

In this context, $\hat{p}$ of the naive algorithm estimates $P_{\mathfrak{S}}(S)$, while p in the new algorithm estimates $P_{\mathfrak{S}|CF}(S)$. Assuming $\hat{p} \to P_{\mathfrak{S}}(S)$ and $p \to P_{\mathfrak{S}|CF}(S)$, the following statements hold:

$$N = \frac{z^2_{1-\alpha/2} \cdot P_{\mathfrak{S}|CF}(S) \cdot \left(1 - P_{\mathfrak{S}|CF}(S)\right)}{\epsilon^2} \cdot P^2_{CF}(S)$$

$$\hat{N} = \frac{z^2_{1-\alpha/2} \cdot P_{\mathfrak{S}}(S) \cdot (1 - P_{\mathfrak{S}}(S))}{\epsilon^2}$$

Therefore we can write N in terms of $\hat{N}$ as follows:

$$N = \hat{N} \cdot \frac{P_{\mathfrak{S}|CF}(S) \cdot \left(1 - P_{\mathfrak{S}|CF}(S)\right)}{P_{\mathfrak{S}}(S) \cdot (1 - P_{\mathfrak{S}}(S))} \cdot P^2_{CF}(S)$$

Since $P_{\mathfrak{S}}(S) = P_{\mathfrak{S}|CF}(S) \cdot P_{CF}(S)$ we derive the following relation:

$$N = \hat{N} \cdot \frac{1 - P_{\mathfrak{S}|CF}(S)}{1 - P_{\mathfrak{S}}(S)} \cdot P_{CF}(S) \tag{3.4}$$

Together with the number of trials, we want to determine the relation between the number of successes $\hat{X}$ of the naive algorithm and the number of successes X of the new algorithm. As we will see in the experiments, this is an interesting measure for the grounded semantics when $P_{complete}(S) \cong P_{grounded}(S)$.

Under the same above assumptions and the basic laws of probability, we can assert what follows:

$$\hat{X} = \hat{N} \cdot P_{\mathfrak{S}}(S) \qquad\qquad X = N \cdot P_{\mathfrak{S}|CF}(S)$$

By substituting equation (3.4) we get the following:

$$X = \hat{N} \cdot \frac{P_{\mathfrak{S}|CF}(S) \cdot \left(1 - P_{\mathfrak{S}|CF}(S)\right)}{P_{\mathfrak{S}}(S) \cdot (1 - P_{\mathfrak{S}}(S))} \cdot P_{CF}^2(S) \cdot P_{\mathfrak{S}|CF}(S)$$

Since $P_{\mathfrak{S}}(S) = P_{\mathfrak{S}|CF}(S) \cdot P_{CF}(S)$ the following holds:

$$X = \hat{N} \cdot P_{\mathfrak{S}}(S) \cdot \frac{1 - P_{\mathfrak{S}|CF}(S)}{1 - P_{\mathfrak{S}}(S)}$$

Finally, from $\hat{X} = \hat{N} \cdot P_{\mathfrak{S}}(S)$ we can derive the following relation between X and $\hat{X}$ (and recall equation (3.4) for completeness):

$$X = \hat{X} \cdot \frac{1 - P_{\mathfrak{S}|CF}(S)}{1 - P_{\mathfrak{S}}(S)} \qquad\qquad N = \hat{N} \cdot \frac{1 - P_{\mathfrak{S}|CF}(S)}{1 - P_{\mathfrak{S}}(S)} \cdot P_{CF}(S)$$

We observe that $P_{\mathfrak{S}|CF}(S) \geq P_{\mathfrak{S}}(S)$, from which $1 - P_{\mathfrak{S}|CF}(S) \leq 1 - P_{\mathfrak{S}}(S)$ follows. Therefore it is easy to see that an upperbound to the number of trials is $N \leq \hat{N} \cdot P_{CF}(S)$.

For what concerns the number of successes, there is no good news because X can assume any value between 0 and 1 independently from $P_{CF}(S)$. In the worst case where $P_{\mathfrak{S}|CF}(S) \to 0$, $X \to \hat{X}$.

Fixed $P_{CF}(S)$, the lower $P_{\mathfrak{S}|CF}(S)$ is the higher the number of successes with respect to the naive algorithm is. This is a surprising result because intuitively $X \propto P_{\mathfrak{S}|CF}(S)$.

Fixed $P_{\mathfrak{S}|CF}(S)$, it is easy to see that $1 - P_{\mathfrak{S}|CF}(S) \leq \frac{1-P_{\mathfrak{S}|CF}(S)}{1-P_{\mathfrak{S}}(S)} \leq 1$, and that the lower $P_{CF}(S)$ is the lower the number of successes with respect to the naive algorithm is.

4 Evaluation of hard cases in probabilistic argumentation frameworks

4.1 Architecture and setting for the experiments

We have described, given some PrAF, two approaches to computing the likelihood of a chosen set of arguments being justified with respect to some semantics. Our purpose is to evaluate the running time and iterations of the optimized Monte Carlo algorithm compared to the naive algorithm, given the same error level.

Both algorithms have been implemented in Java by using hashed data structures to efficiently encode the sparseness of the graphs.

The experiments have been run for differently sized PrAFs containing between 10 and 20 arguments, the chosen sets of arguments containing between 6 and 8 arguments and an error level of $\epsilon = 0.01$ and $\epsilon = 0.005$ with confidence level of 95%.

First, we randomly generate a PrAF with 20 arguments given three probability distributions: one over argument probabilities, one over the outdegree of arguments and one over the defeat probabilities.

Second, we search for a suitable set of arguments S for the generated PrAF in order to satisfy the following constraints:

- $0.6 \leq P_{CF}(S) < 1.0$, to obtain a higher $P_{\mathfrak{S}}(S)$;
- $P_{\mathfrak{S}}(S) \geq 0.2$, otherwise the resulting probabilities wouldn't be perceptible.

We repeat the process of generating the instance $\langle PrAF, S \rangle$ for a number of iterations, then we choose the most satisfying based on the above constraints.

Third, we delete arguments from this $\langle PrAF, S \rangle$ instance in order to generate the remaining instances with a number of arguments between 10 and 19, yet satisfying the above constraints.

Finally, given an error level of either $\epsilon = 0.01$ or $\epsilon = 0.005$ and confidence level of 95% ($z_{1-\alpha/2} = 1.96$), for each semantics $\mathfrak{S}$ and instance $\langle PrAF, S \rangle$ we run our experiment 20 times, that is we compute $P_{\mathfrak{S}}(S)$ with both algorithms to estimate the CPU time, the number of successes and the number of iterations.

4.2 Generating probabilistic argumentation frameworks

We randomly generate a $PrAF = \langle A, P_A, D, P_D \rangle$ given a number of arguments and the probability distributions shown in figure 4.1.

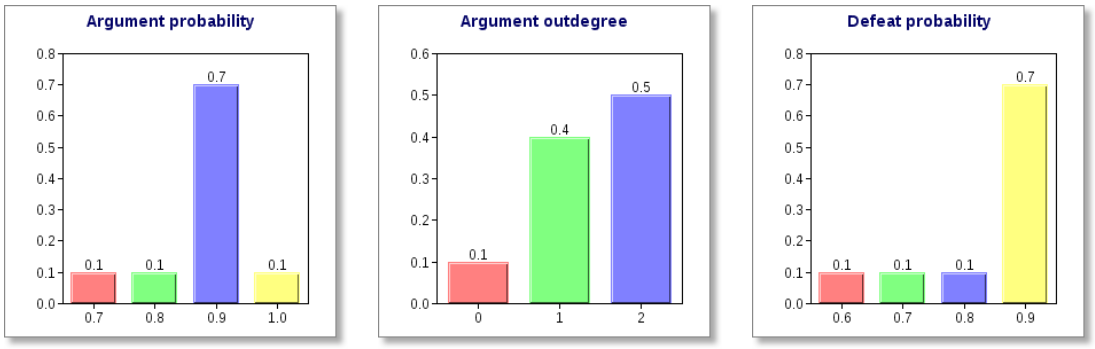


Figure 4.1: Probability distributions for generating PrAFs

We used high probabilities for arguments in order to increase $P_{CF}(S)$ of a possible set S. For example, if $|S| = 8$ and $\forall a \in S\ P_A(a) = 0.95$ and there's no defeat between arguments in S, then $P_{CF}(S) = (0.95)^8 \cong 0.66$ which already near the lower bound.

The outdegree of arguments instead is low because a higher number would increase the chances to have defeats between arguments in S and thus make $P_{CF}(S)$ even lower.

About the defeat probabilities it might seem counterintuitive to have an high probability, but it's required to have higher $P_{complete}(S)$ and consequentially a higher $P_{\mathfrak{S}}(S)$ in case of complete and grounded semantics. For instance, in figure 4.2a it will be highly probable that the defeat will not occur and then the set S will not complete because b is acceptable. In figure 4.2b instead the defeat will likely occur so that $P_{complete}(S)$ will be higher together with $P_{\mathfrak{S}}(S)$.

4.3 Searching for a suitable set of arguments

Given PrAF we choose a set of arguments S such that $0.6 \leq P_{CF}(S) < 1.0$ and $P_{\mathfrak{S}}(S)$ to be as high as we can in order to obtain perceptible results. This problem

(a) Lower $P_{complete}(S)$ (b) Higher $P_{complete}(S)$

Figure 4.2: Example of probabilities over defeats

is harder than the original problem, because not only we have to evaluate $P_{\mathfrak{S}}(S)$ but also find the set S.

Given a $PrAF = \langle A, P_A, D, P_D \rangle$ we find the set S such that it maximizes the following objective function:

$$\arg\max_S c_1 \cdot \prod_{a \in S} P_A(a) - c_2 \cdot \sum_{\substack{a \in S \\ b \in A}} P_A(b) \cdot P_D(b,a) - c_3 \cdot \sum_{a \notin S} P_A(a)$$

Intuitively, the first term we maximize is the event that S occurrs so that $P_{CF}(S)$ will be high. The second term minimizes the probability of attacks against the set S so that $P_{admissible}(S)$ will be high. Then we minimize the event that arguments outside of S occurs so that $P_{complete}(S)$ will be high.

We are not interested in the final value of the function, so we decide to substitute the first term with a sum yet have similar results for the set S. The objective function is now linear and S can be expressed with binary variables s_a such that $a \in S$ iff $s_a = 1$:

$$\arg\max_S c_1 \cdot \sum_{a \in A} s_a \cdot P_A(a) - c_2 \cdot \sum_{a,b \in A} s_a \cdot P_A(b) \cdot P_D(b,a) - c_3 \cdot \sum_{a \in A} (1 - s_a) \cdot P_A(a)$$
$$\forall a \in A : \; s_a \in \{0,1\}$$

The three constants have been chosen empirically to be: $c_1 = 1, c_2 = 3, c_3 = 1$.

After finding S by solving the above mixed-integer linear programming problem, it is possible to run local search algorithms in order to improve both $P_{CF}(S)$ and $P_{\mathfrak{S}}(S)$. We adopted a simple hill climbing by generating the neighborhood N_S of S with three mutation rules:

$$N_S = \left\{ \hat{S} \mid \begin{array}{c} \hat{S} = S \setminus \{a\}, a \in S \vee \\ \hat{S} = S \cup \{a\}, a \notin S \vee \\ \hat{S} = (S \setminus \{a\}) \cup \{b\}, a \in S, b \notin S \end{array} \right\}$$

This will usually lead to a set of arguments S with $0.6 \leq P_{CF}(S) < 1.0$ but very low $P_{\mathfrak{S}}(S)$. For this reason, we manually manipulate the probabilities on arguments and defeats to increase $P_{\mathfrak{S}}(S)$ yet keeping $P_{CF}(S)$ constrained. This is doable as long as the number of arguments is small.

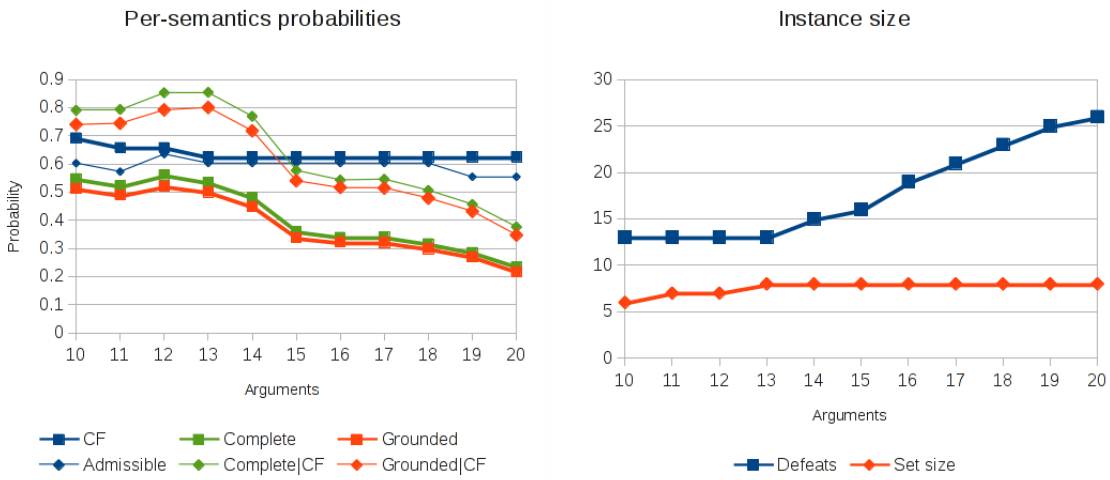
4.4 Results

In the previous sections we constructed 10 instances $\langle PrAF, S \rangle$ of the problem, with number of arguments between 10 and 20.

The charts in figure 4.3 describe the characteristics of each problem instance in terms of size and probabilities for each semantics computed with an error of $\epsilon = 0.005$. We intentionally let the values of $P_{complete}(S)$ and $P_{grounded}(S)$ decrease with the number of arguments and defeats, otherwise the problem of finding a better set S would be too hard.

The cardinality of the chosen set of arguments is $|S| = 6$ for the the $PrAF$ of 10 arguments, $|S| = 7$ for 11 and 12 arguments, and $|S| = 8$ for $|A| \geq 13$.

For 10 arguments $P_{CF}(S) = 0.69$ while for 20 arguments $P_{CF}(S) = 0.62$.



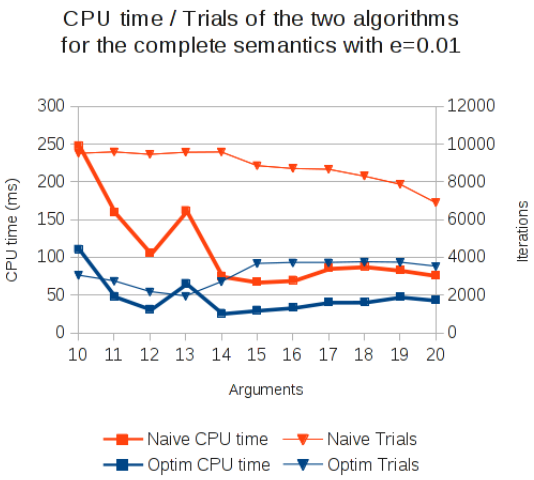
(a) Probability of each semantics

(b) Defeats and arguments in S

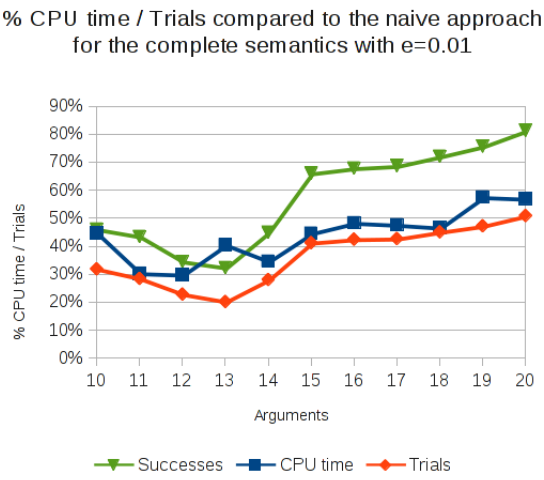
Figure 4.3: Characteristics of chosen problem instances

We chose to run the experiments for the complete and grounded semantics. For each instance and for each semantics we run the experiment 20 times, with a confidence level of 95% and error levels of $\epsilon = 0.01$ and $\epsilon = 0.005$. Finally we compute the mean value of the CPU time and the percentage of the number of Monte Carlo trials and successes with respect to the naive algorithm.

The run time comparison for the complete semantics is shown in figure 4.4 with $\epsilon = 0.01$ and in figure 4.5 with $\epsilon = 0.005$. As expected the number of iterations for 20 arguments is $N \leq P_{CF}(S) = 0.62$.

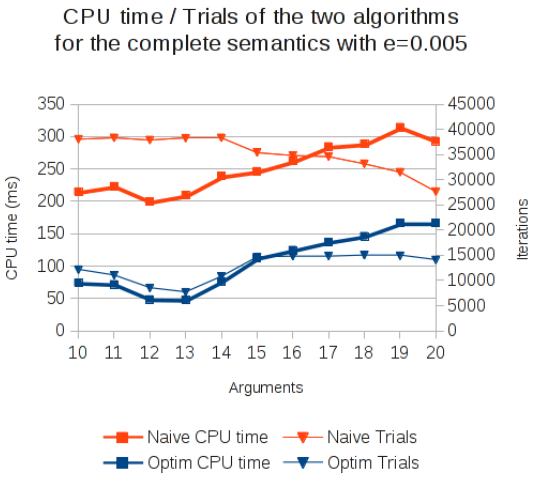


(a) Absolute time and number of trials

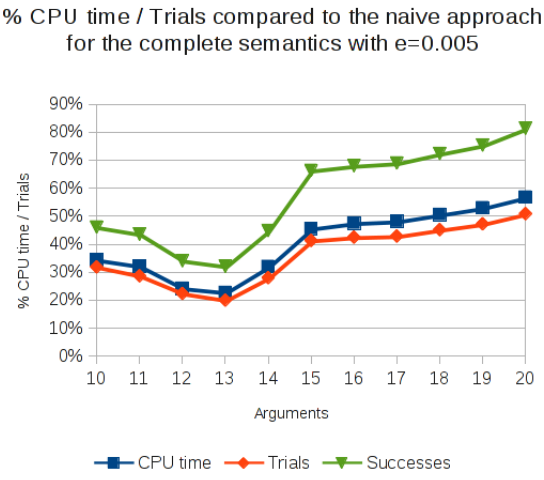


(b) Time, trials and successes percentage

Figure 4.4: Run time comparison for the complete semantics with $\epsilon = 0.01$



(a) Absolute time and number of trials



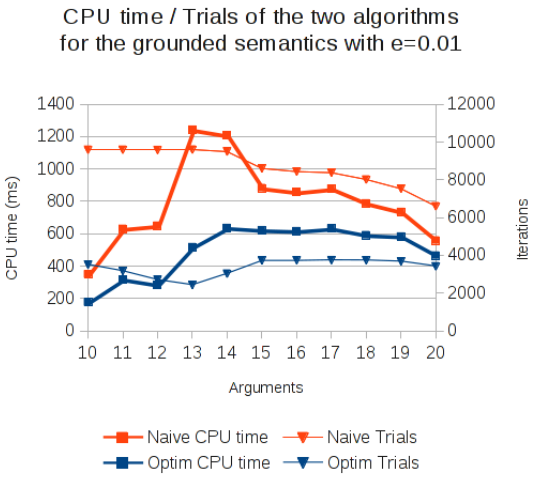
(b) Time, trials and successes percentage

Figure 4.5: Run time comparison for the complete semantics with $\epsilon = 0.005$

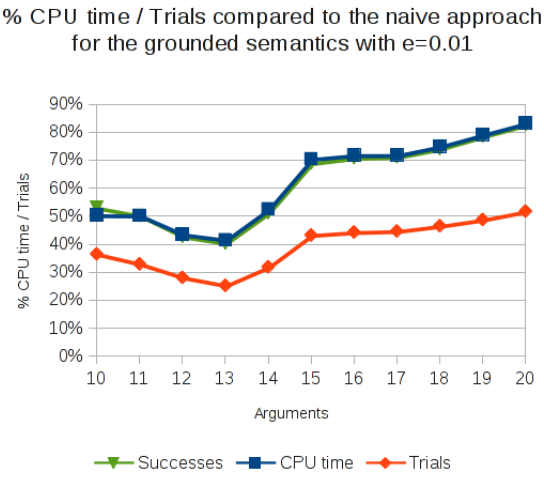
Deciding whether a set of arguments is complete can be computed in polynomial time with respect to the number of arguments and defeats for a given AAF. For

this reason, it can be seen that there is a constant relation between the number of iterations and the CPU time percentage for the complete semantics, that is more especially appreciable in figure 4.5b.

This is not true for the grounded semantics which is an harder decision problem as shown in figure 4.6 with $\epsilon = 0.01$ and figure 4.7 with $\epsilon = 0.005$. Interestingly, the CPU time is very tied to the number of successes with respect to the naive algorithm. The reason is that, in our experiments, $P_{grounded}(S) \cong P_{complete}(S)$, therefore S is unlikely to be complete but not grounded in any induced AAF.

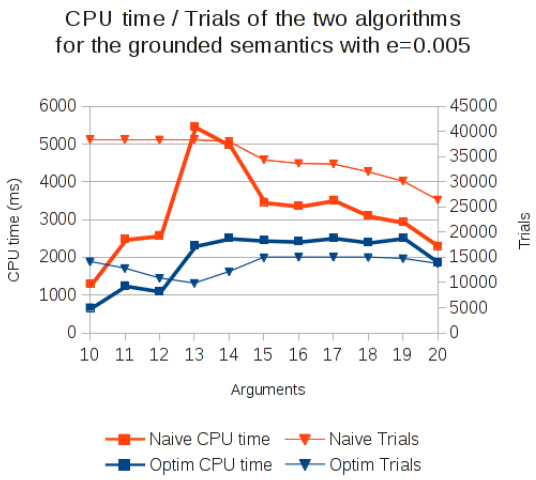


(a) Absolute time and number of trials

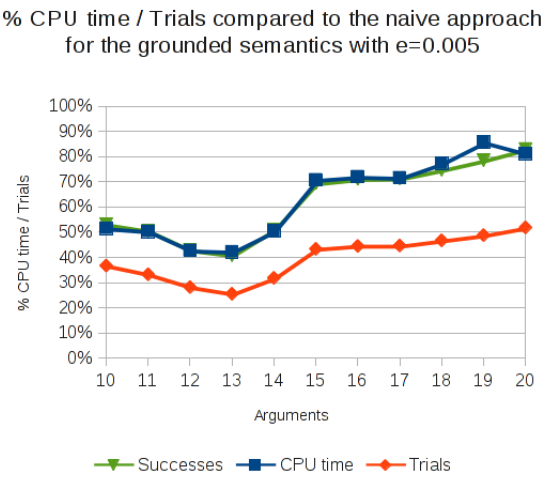


(b) Time, trials and successes percentage

Figure 4.6: Run time comparison for the grounded semantics with $\epsilon = 0.01$



(a) Absolute time and number of trials



(b) Time, trials and successes percentage

Figure 4.7: Run time comparison for the grounded semantics with $\epsilon = 0.005$

As we have seen the new algorithm, based on the novel concept that the probability of a set of arguments being conflict-free is computable in polynomial time, is generally faster than the naive algorithm. In particular, under the assumption that for the complete semantics the CPU time is strictly related to the number of iterations, it follows that the new algorithm is at least $1 - P_{CF}(S)$ faster compared to the naive algorithm.

5 Conclusions and future work

We have described abstract argumentation frameworks and probabilistic argumentation frameworks. An existing approach based on Monte Carlo simulation to computing the probability of a set of arguments being consistent according to some semantics has been evaluated.

We have shown that the probability of a set of arguments being conflict-free and admissible in PrAFs can be computed in polynomial time with respect to the number of arguments and defeats. Then we have proposed a new algorithm that

extends the naive Monte Carlo simulation to take advantage of the novel concept that the probability of a set being conflict-free is computable in polynomial time. Finally, we have shown that the new algorithm is generally more efficient compared to the naive algorithm, and is especially efficient in the case of complete semantics.

This work can be extended to the case of admissibility, whereas the probability of a set of arguments being admissible is also computable in polynomial time. In this case, the problem of sampling the space of admissible AAFs that can be induced from a PrAF is not as trivial as that of conflict-free AAFs.

References

[1] Alan Agresti and Brent A. Coull. Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, 52(2):119–126, 1998.

[2] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93(1-2):63–101, June 1997.

[3] Lawrence D. Brown, T. Tony Cai, and Anirban Dasgupta. Interval estimation for a binomial proportion. *Statistical Science*, 16:101–133, 2001.

[4] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[5] Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. In *Argumentation in Artificial Intelligence*, pages 85–104. 2009.

[6] Xiuyi Fan and Francesca Toni. Conflict resolution with argumentation dialogues. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1095–1096, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[7] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Commun. ACM*, 55(3):106–113, March 2012.

[8] Hengfei Li, Nir Oren, and Timothy J. Norman. Probabilistic argumentation frameworks. In *Proceedings of the First international conference on Theory and Applications of Formal Argumentation*, TAFA'11, pages 1–16, Berlin, Heidelberg, 2012. Springer-Verlag.

[9] David Maxwell Chickering and David Heckerman. Efficient approximations for the marginallikelihood of bayesian networks with hidden variables. *Mach. Learn.*, 29(2-3):181–212, November 1997.

[10] James F. Peters and Christopher Henry. Reinforcement learning with approximation spaces. *Fundam. Inf.*, 71(2,3):323–349, February 2006.

[11] Tjitze Rienstra. Towards a probabilistic dung-style argumentation system. In *Proc. of the First International Conference on Agreement Technologies (AT)*, pages 138–152, 2012.

[12] Matthias Thimm. A probabilistic semantics for abstract argumentation. In *ECAI*, pages 750–755, 2012.

[13] Jinping Yuan, Li Yao, Zhiyong Hao, Fang Liu, and Tangming Yuan. Multi-party dialogue games for distributed argumentation system. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '11, pages 329–332, Washington, DC, USA, 2011. IEEE Computer Society.