# An Extension-based Approach for Computing and Verifying Preferences in Abstract Argumentation

Quratul-ain Mahesar[a,∗], Nir Oren[b], Wamberto W. Vasconcelos[b]

[a]*School of Computing and Engineering, University of Huddersfield, UK*
[b]*Department of Computing Science, University of Aberdeen, UK*

## Abstract

We present an extension-based approach for computing and verifying preferences in an abstract argumentation system. Although numerous argumentation semantics have been developed previously for identifying acceptable sets of arguments from an argumentation framework, there is a lack of justification behind their acceptability based on implicit argument preferences. Preference-based argumentation frameworks allow one to determine what arguments are justified given a set of preferences. Our research considers the inverse of the standard reasoning problem, i.e., given an abstract argumentation framework and a set of justified arguments, we compute what the possible preferences over arguments are. Furthermore, there is a need to verify (i.e., assess) that the computed preferences would lead to the acceptable sets of arguments. This paper presents a novel approach and algorithm for exhaustively computing and enumerating all possible sets of preferences (restricted to three identified cases) for a conflict-free set of arguments in an abstract argumentation framework. We prove the soundness, completeness and termination of the algorithm. The research establishes that preferences are determined using an extension-based approach after the evaluation phase (acceptability of arguments) rather than stated beforehand. In this work, we focus our research study on grounded, preferred and stable semantics. We show that the complexity of computing sets of preferences is exponential in the number of arguments, and thus, describe an approximate approach and algorithm to compute the preferences. Furthermore, we present novel algorithms for verifying (i.e., assessing) the computed preferences. We provide details of the implementation of the algorithms (source code has been made available), various experiments performed to evaluate the algorithms and the analysis of the results.

*Keywords:* Abstract Argumentation, Preferences, Reasoning

## 1. Introduction

Preferences play a central part in decision making and have been extensively studied in various disciplines such as economy, operations research, psychology and philosophy [1]. Preferences are used in many areas of artificial intelligence including planning, scheduling, multi-agent systems, combinatorial auctions and game playing [2]. Preference elicitation is a very difficult task and automating the process of preference extraction can be very difficult. The complexity of eliciting preferences and representational questions like dealing with uncertainty has remained a very active research area [3, 4, 2]. Preference elicitation plays a vital role in decision support systems [5, 6] and recommender systems [7], where the most suitable decision(s) or recommendation(s) can be identified and justified with the help of preferences. Furthermore, elicited preferences can be utilized in dialogue strategies [8, 9], for instance in computational persuasion [10, 11] or negotiation [12] – where an agent may have the capability of inferring preferences and reach her goal if (s)he enforces at least one of several desired sets of arguments with the application of preferences. The inferred preferences can be exploited in optimizing the choice of move in persuasion dialogues for behaviour change as well as in negotiation dialogues to reach agreement.

---

∗Corresponding author
*Email addresses:* `q.mahesar@hud.ac.uk` (Quratul-ain Mahesar), `n.oren@abdn.ac.uk` (Nir Oren), `w.w.vasconcelos@abdn.ac.uk` (Wamberto W. Vasconcelos)

Argumentation has gained an increasing popularity in Artificial Intelligence (AI). It has been widely used for handling inconsistent knowledge bases [13, 14, 15], and dealing with uncertainty in decision making [16, 17, 18]. Logic-based abstract argumentation [19] provides a formal representation of preferences. An abstract argumentation framework is a directed graph consisting of nodes that represent unique atomic arguments and directed edges that represent an attack between two arguments. This visual representation of an argumentation framework as a directed graph is also known as an argumentation graph. Acceptable sets of arguments called extensions for an argumentation framework can be computed based on various acceptability semantics [19].

Arguments can have different strengths, e.g., an argument relies on more certain or important information than another. This has led to the introduction of preference-based argumentation frameworks consisting of preference relations between arguments [20, 16, 21, 22]. Furthermore, preferences are taken into account in the evaluation of arguments at the semantic level, which is also known as preference-based acceptability [23]. The basic idea is to accept undefeated arguments and also arguments that are preferred to their attacking arguments, as these arguments can defend themselves against their attacking arguments.

Preference-based argumentation framework (PAF) [20, 24] allows one to determine what arguments are justified given a set of preferences. In our research, we consider the inverse of the standard reasoning problem, i.e., given an abstract argumentation framework and a set of justified arguments, we compute what the possible preferences over arguments are. Although a preference-based argumentation framework (PAF) has been previously studied to represent an abstract argumentation framework [25], there seems to be no previous work on automatically computing implicit argument preferences in an abstract argumentation framework using an extension-based approach. Furthermore, there have been no attempts to perform an exhaustive search for all possible preferences, and their explicit enumeration.

There are two aims of our research study. The first aim of our research is to exhaustively compute all possible sets of argument preferences (restricted to three identified cases) that hold for a given set of conflict-free arguments, i.e., extension, in an abstract argumentation framework. In this work, we focus our research study on grounded, preferred and stable semantics. We present a novel algorithm to perform this computation. We show that the complexity of computing sets of preferences is exponential in the number of arguments, and thus, describe an approximate approach and algorithm to compute the preferences that is scalable. The second aim of our research is to verify (i.e., assess) that all the computed sets of preferences are correct, i.e., each set of preferences when applied to a given abstract argumentation framework results in the original input extension under a given semantics. We present novel algorithms to perform this verification. All algorithms have been implemented. We have build a complete system for computing and verifying preferences, performed various experiments to evaluate the algorithms and analyze the results.

The current paper extends and improves our previous work [26]. The main contributions of our work are as follows:

1. An extension-based approach is employed for computing and verifying argument preferences. Thus, preferences specifically justify the reasoning behind the acceptability of the arguments in an extension.

2. Preferences are computed at the end of the argumentation process and need not be stated in advance.

3. Exhaustive search is performed to compute all possible sets of preferences.

4. The approach for computing preferences operates on a conflict-free extension as input which is the minimal acceptability semantic, therefore, it can take as input most of the extensions given in the literature and stated in this paper.

5. We present novel algorithms for computing preferences (and additional algorithms for filtering preferences).

6. We present a novel approximate algorithm for computing preferences.

7. We present novel algorithms for verifying preferences.

8. Reference implementation of our algorithms is provided[1].

---

9. Experimental setup (including data sets in the Appendix[2]), various experiments that have been performed to evaluate the algorithms and analysis of the results obtained is presented.

In comparison to [26], the contributions 6, 7, 8 and 9 stated above are new.

The remainder of this paper is structured as follows. In Section 2, we present related work. Section 3 presents the preliminaries that include the background on abstract argumentation framework and acceptability semantics for acceptable set of arguments also known as extensions. This is followed by background on preference-based argumentation framework. In Section 4, we present our approach and an algorithm for computing all possible sets of preferences for a given extension and abstract argumentation framework, and we prove the soundness, completeness and termination of the algorithm. Additionally, we present algorithms for filtering preferences. Furthermore, we present an approximate approach and algorithm for computing a set of preferences. In Section 5, we present our approach and algorithms for verifying all computed sets of preferences for a given extension and abstract argumentation framework. In Section 6, we present the implementation details and evaluation. Finally, we conclude and suggest future work in Section 7.

## 2. Related Work

Several variations of argumentation frameworks with preferences have been studied previously. Preference-based argumentation framework (PAF) [20, 24] is an extension of a standard argumentation framework [19] consisting of preference relations between arguments. The idea is to accept undefeated arguments and also arguments that are preferred to their attacking arguments. Value-based argumentation framework (VAF) [27] extends a standard argumentation framework to take into account values promoted by arguments. Preferences over arguments are determined by the values the arguments promote or support. The idea is to accept undefeated arguments and also arguments which promote values that are more important or preferred to the values promoted by their attacking arguments. Furthermore, value-based argumentation frameworks (VAF) have been extended to take into account the possibility that arguments may support multiple values, and therefore, various types of preferences over values could be considered in order to deal with real world situations [25]. Another variation is an extended argumentation framework (EAF) [21] that considers the case where arguments can express preferences between other arguments.

Further studies on preference-based argumentation frameworks led to the observation that ignoring the attacks where the attacked argument is stronger than the attacking argument does not always give intuitive results [21], since the resulting extension violates the basic condition imposed on acceptability semantics, which is the conflict-freeness of extensions, thus violating the rationality postulates given in [28]. This problem was later resolved in a new preference-based argumentation framework that guarantees conflict-free extensions with a symmetric conflict relation [29, 21]. The preference relation is then used to determine the direction of the defeat relation between the two arguments. Furthermore, preference relations have been used to refine the results of a framework by comparing its extensions [24].

Although our work is based on abstract argumentation framework, several variations of structured argumentation frameworks with preferences have been studied previously. ABA$^+$ [30] generalises preference-based argumentation framework (PAF) [24] that introduced the concept of attack reversal from less preferred arguments. Another extension of the ABA framework with preferences is (p_ABA) [31] that employs preferences on the extension level to discriminate among extensions. ASPIC$^+$ [32] encompasses many key elements of structured argumentation such as strict and defeasible rules, general contrariness mapping and various forms of attacks as well as preferences. DeLP [14], an early version of preference-based argumentation framework [24], and Deductive Argumentation [33], use preferences to discard attacks from arguments less preferred than the attacked arguments.

While the above argumentation frameworks allow handling of preferences over arguments, the main limitation is that preferences need to be stated in advance. More recently, in [34] we have extended our work [26] in abstract argumentation to structured argumentation, i.e., assumption-based argumentation frameworks ABA [35, 36] and ABA$^+$ [30], where preferences are computed at the assumption level rather than abstract arguments. Another recent extension is presented in [37], where hidden argument preferences of a group of agents are revealed using

---

[2] All data sets are included in the Appendix.

answer sets. Furthermore, to the best of our knowledge, no previous work has been performed for the verification of the newly computed preferences along with the implementation and experimental analysis of both the computation and verification of preferences, that is presented in this extended version of our paper [26].

## 3. Preliminaries

An *argumentation framework* is a set of arguments and a binary attack relation among them. Given an argumentation framework, argumentation theory allows to identify the sets of arguments that can survive the conflicts expressed in the framework. In this work, we consider finite abstract argumentation frameworks.

**Definition 3.1.** *(Abstract Argumentation Framework [19]): An abstract argumentation framework (AAF) is a pair $AAF = (\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$). The notation $(A, B) \in \mathcal{R}$ where $A, B \in \mathcal{A}$ denotes that A attacks B.*
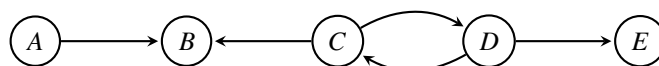


Figure 1: Example abstract argumentation framework $AAF_1$

An abstract argumentation framework is a directed graph where the arguments are represented as nodes and the attack relations as directed edges. An example abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ is shown in Figure 1, where $\mathcal{A} = \{A, B, C, D, E\}$ and $\mathcal{R} = \{(A, B), (C, B), (C, D), (D, C), (D, E)\}$, which means that $A$ attacks $B$, $C$ attacks both $B$ and $D$, and $D$ attacks both $C$ and $E$.

Dung [19] originally introduced an extension approach to define the acceptability of arguments in an argumentation framework. An extension is a subset of $\mathcal{A}$ that represents the set of arguments that can be accepted together. Dung's semantics are based on a *conflict-free* set of arguments, i.e., a set should not be self-contradictory nor include arguments that attack each other. This is defined formally as follows.

**Definition 3.2.** *(Conflict-freeness): Let $(\mathcal{A}, \mathcal{R})$ be an argumentation framework. The set $\mathcal{E} \subseteq \mathcal{A}$ is conflict-free if and only if there are no $A, B \in \mathcal{E}$ such that $(A, B) \in \mathcal{R}$*

The minimal requirement for an extension to be acceptable is *conflict-freeness*. Many other acceptability semantics have been introduced in the literature, and from these the most common are given as follows.

**Definition 3.3.** *(Extensions): Let $AAF = (\mathcal{A}, \mathcal{R})$ be an argumentation framework, and set $\mathcal{E} \subseteq \mathcal{A}$ and $A, B, C \in \mathcal{A}$*

- $\mathcal{E}$ *is admissible iff it is conflict free and defends all its arguments. $\mathcal{E}$ defends A iff for every argument $B \in \mathcal{A}$, if we have $(B, A) \in \mathcal{R}$ then there exists $C \in \mathcal{E}$ such that $(C, B) \in \mathcal{R}$.*

- $\mathcal{E}$ *is a complete extension iff $\mathcal{E}$ is an admissible set which contains all the arguments it defends.*

- $\mathcal{E}$ *is a preferred extension iff it is a maximal (with respect to set inclusion) admissible set.*

- $\mathcal{E}$ *is a stable extension iff it is conflict-free and for all $A \in \mathcal{A} \setminus \mathcal{E}$, there exists an argument $B \in \mathcal{E}$ such that $(B, A) \in \mathcal{R}$.*

- $\mathcal{E}$ *is a grounded extension iff $\mathcal{E}$ is a minimal (for set inclusion) complete extension.*

Every argumentation framework has at least one admissible set (the empty set), exactly one grounded extension, one or more complete extensions, one or more preferred extensions, and zero or more stable extensions. The following example shows the extensions for the abstract argumentation framework of Figure 1.

**Example 3.1.** *Given the abstract argumentation framework of Figure 1, then we compute its extensions as follows:*

- *Conflict free:* $\{A, C, E\}, \{A, D\}, \{B, D\}, \{A, C\}, \{A, E\}, \{B, E\}, \{C, E\}, \{A\}, \{B\},$
  $\{C\}, \{D\}, \{E\}, \emptyset$

- *Admissible:* $\{A, C, E\}, \{A, C\}, \{A, D\}, \{C, E\}, \{A\}, \{C\}, \{D\}, \emptyset$

- *Complete:* $\{A, C, E\}, \{A, D\}, \{A\}$

- *Preferred:* $\{A, C, E\}, \{A, D\}$

- *Stable:* $\{A, C, E\}, \{A, D\}$

- *Grounded:* $\{A\}$

While an abstract argumentation framework captures the basic interactions between arguments, it does not consider factors such as argument strength, i.e., arguments may not necessarily have the same strengths [38, 39, 15]. Consequently, preferences over arguments can be added to the argumentation framework and taken into account in order to evaluate arguments [16, 21, 22], which is demonstrated in the following example [23].

**Example 3.2.** *Let $(\mathcal{A}, \mathcal{R})$ be an argumentation framework with $\mathcal{A} = \{A, B, C\}$ and $\mathcal{R} = \{(A, B), (B, C)\}$. The set of acceptable argument is $\{A, C\}$. However, suppose argument B is preferred to A and C. How can we combine the preference over arguments and the attack relation to decide which arguments are acceptable? We can say that, since B is preferred to A, it can defend itself from the attack of A. This would lead us to accepting B and rejecting C.*

Dung's framework has been extended by introducing preference relations into argumentation systems, which is known as a preference-based argumentation framework (PAF) [20]. A PAF extends an abstract argumentation framework to account for preferences over arguments. The attack relation in a preference-based argumentation framework is called defeat, and is denoted by *Def*.

**Definition 3.4.** *(Preference-based Argumentation Framework (PAF) [20]): A preference-based argumentation framework is a triple $(\mathcal{A}, Def, \geq)$ where $\mathcal{A}$ is a set of arguments, Def is the defeat binary relation on $\mathcal{A}$, and $\geq$ is a (partial or total) pre-ordering defined on $\mathcal{A} \times \mathcal{A}$. The notation $(A, B) \in Def$ means that argument A defeats argument B.*

The notation $A \geq B$ means that argument $A$ is at least as preferred as $B$ and the relation $>$ is the strict counterpart of $\geq$.

**Definition 3.5.** *Let there be an abstract argumentation framework. Preferences could be applied in two ways [24]:*

1. *one way is to apply preferences at the time of argument acceptability (semantic level); and*

2. *second way is to compute all preferred extensions and filter them by the application of the preferences.*
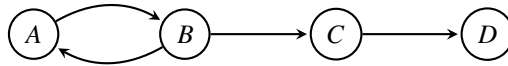


Figure 2: Example abstract argumentation framework $AAF_2$

**Example 3.3.** *Let there be an abstract argumentation framework of Figure 2. By using the first method given in Definition 3.5, if we assume $\{A > B, C > D\}$ is the set of preferences between arguments, then we get a single extension $\mathcal{E} = \{A, C\}$. Now, by using the second method, we first compute all preferred extensions $\{A, C\}, \{B, D\}$. These extensions could now be filtered by the application of the set of preferences $\{A > B, C > D\}$ which suggest $\{A, C\}$ to be better than $\{B, D\}$.*

In this work, we use the first method of applying preferences at the time of argument acceptability (semantic level).

## 4. Computing Preferences

A preference-based argumentation framework (PAF) can represent an abstract argumentation framework [25]:

**Definition 4.1.** *(PAF representing an AAF) A preference-based argumentation framework $(\mathcal{A}, Def, \geq)$ represents an abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ iff $\forall A, B \in \mathcal{A}$, it is the case that $(A, B) \in \mathcal{R}$ iff $(A, B) \in Def$ and it is not the case that $B > A$.*

It has been previously shown that each preference-based argumentation framework represents one abstract argumentation framework, however each abstract argumentation framework can be represented by various preference-based argumentation frameworks [25]. Following this, we introduce an extension-based approach for computing sets of preferences for a subset of conflict-free arguments in an abstract argumentation framework. For any two arguments $A$ and $B$ in an argumentation framework, we use the strict preference relation $A > B$ to denote that $A$ is strictly preferred to $B$, i.e., $A$ is of greater strength than $B$, and we use the preference relation $A = B$ to denote that $A$ and $B$ are of equal strength or preference. We list below the three cases we have identified for which the preferences are computed for a given conflict-free extension $\mathcal{E}$ in an abstract argumentation framework $AAF = \langle \mathcal{A}, \mathcal{R} \rangle$. The motivation behind the identified three cases is that we want to find out why a set of arguments are in an extension of the abstract argumentation framework based on the relationship of attack relations between the arguments and their strengths (i.e., preferences between the arguments). The three identified cases are:

- **Case 1:** Suppose $\alpha, \beta \in \mathcal{A}$ and $\alpha \in \mathcal{E}, \beta \notin \mathcal{E}$ such that $\alpha$ is attacked by argument $\beta$, and $\alpha$ is not defended by any other argument (not equal to $\alpha$) in the extension. We have the following preferences for all such $\alpha$ and $\beta$: $\alpha > \beta$.

- **Case 2:** Suppose $\alpha, \beta \in \mathcal{A}$ and $\alpha \in \mathcal{E}, \beta \notin \mathcal{E}$, and suppose $\alpha$ attacks argument $\beta$ and $\beta$ does not attack $\alpha$. We have the following preferences for all such $\alpha$ and $\beta$: $\beta \not> \alpha$, i.e., $(\alpha > \beta) \vee (\alpha = \beta)$.

- **Case 3:** Suppose $\alpha, \beta, \gamma \in \mathcal{A}$ and $\alpha, \gamma \in \mathcal{E}, \beta \notin \mathcal{E}$ where $\alpha, \beta$ and $\gamma$ are different arguments, such that, $\alpha$ is attacked by argument $\beta$ but defended by argument $\gamma$ in the extension, i.e., $\gamma$ attacks $\beta$. We have the following preferences for all such $\alpha$ and $\beta$: $(\alpha > \beta) \vee (\alpha = \beta) \vee (\beta > \alpha)$.

In other words, we want to determine and compute the preferences between arguments that will ensure that a set of desired arguments are in an extension of the abstract argumentation framework by establishing that:

- the attacks from arguments that are not in the extension to the arguments that are in the extension that are not defended by any unattacked arguments in the extension do not succeed, as given in Case 1;

- the attacks from arguments that are in the extension to the arguments that are not in the extension always succeed, as given in Case 2; and

- the attacks from arguments that are not in the extension to the arguments that are in the extension that are defended by any unattacked arguments in the extension may or may not succeed, as given in Case 3.

A worked example of how preferences are computed using the above three cases is as follows:

**Example 4.1.** *Let there be the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ of Figure 1, where $\mathcal{A} = \{A, B, C, D, E\}$ and $\mathcal{R} = \{(A, B), (C, B), (C, D), (D, C), (D, E)\}$. We consider the conflict-free extensions $\mathcal{E}_1 = \{A, C, E\}$, and $\mathcal{E}_2 = \{A, D\}$ for computing preferences. For the extension $\mathcal{E}_1 = \{A, C, E\}$, we have the following preferences for each case:*

- *Case 1:* $(C > D)$

- *Case 2:* $((A > B) \vee (A = B)) \wedge ((C > B) \vee (C = B))$

- *Case 3:* $(E > D) \vee (E = D) \vee (D > E)$

*Combining the preferences from the three cases we get* $(C > D) \wedge (((A > B) \vee (A = B)) \wedge ((C > B) \vee (C = B))) \wedge ((E > D) \vee (E = D) \vee (D > E))$, *which gives us the following sets of preferences:*

$$\{C > D, A > B, C > B, E > D\}$$
$$\{C > D, A > B, C > B, E = D\}$$
$$\{C > D, A > B, C > B, D > E\}$$
$$\{C > D, A > B, C = B, E > D\}$$
$$\{C > D, A > B, C = B, E = D\}$$
$$\{C > D, A > B, C = B, D > E\}$$
$$\{C > D, A = B, C > B, E > D\}$$
$$\{C > D, A = B, C > B, E = D\}$$
$$\{C > D, A = B, C > B, D > E\}$$
$$\{C > D, A = B, C = B, E > D\}$$
$$\{C > D, A = B, C = B, E = D\}$$
$$\{C > D, A = B, C = B, D > E\}$$

*For the extension* $\mathcal{E}_2 = \{A, D\}$, *we have the following preferences for each case:*

- *Case 1:* $(D > C)$

- *Case 2:* $((A > B) \vee (A = B)) \wedge ((D > E) \vee (D = E))$

- *Case 3:* $\emptyset$

*Combining the preferences from the three cases we get* $(D > C) \wedge (((A > B) \vee (A = B)) \wedge ((D > E) \vee (D = E))) \wedge \emptyset$, *which gives us the following sets of preferences:*

$$\{D > C, A > B, D > E\}$$
$$\{D > C, A > B, D = E\}$$
$$\{D > C, A = B, D > E\}$$
$$\{D > C, A = B, D = E\}$$

## 4.1. Algorithms for Computing Preferences

As stated previously, in our research study, we consider the inverse of the standard reasoning problem. We now state the problem of computing preferences precisely as follows.

**Problem 4.1.** *Given an abstract argumentation framework AAF and a single set of justified arguments, i.e., an extension $\mathcal{E}$ under a given semantics (grounded, preferred or stable), we compute what the possible preferences over arguments are, i.e., a set of sets of preferences PrefSet, such that each set of preferences Prefs ∈ PrefSet when applied to the AAF results in the single input extension $\mathcal{E}$ under a given semantics (grounded, preferred or stable)[3].*

To solve Problem 4.1, we present Algorithm 1 that performs the computation of preferences over arguments with the help of Algorithms 2, 3, and 4. We will now present and describe all Algorithms 1, 2, 3, and 4. Algorithm 1 exhaustively computes all possible sets of preferences for a given input extension (consisting of conflict-free arguments) in an abstract argumentation framework (AAF) using the above three cases. The input of Algorithm 1 is a tuple $\langle AAF, \mathcal{E} \rangle$, where:

- Abstract argumentation framework $AAF = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{A}$ denotes the set of all arguments in the $AAF$, and $\mathcal{R}$ denotes the attack relation between arguments.

---

[3]Please note, the application of preferences on an AAF would restrict the result to a single extension $\mathcal{E}$ under multi-extension semantics (i.e., preferred and stable) and single extension semantics (i.e., grounded).

- Extension $\mathcal{E}$ consists of a finite number of conflict-free arguments such that $\mathcal{E} \subseteq \mathcal{A}$.

The algorithm computes and outputs a set consisting of finite sets of preferences, where each set of preferences is represented as $Prefs = \{A > B, B = C, ....\}$ such that $\{A, B, C, ...\} \subseteq \mathcal{A}$. The following are the main steps in Algorithm 1:

- Line 2: Invoke Algorithm 2 with inputs $AAF$ and $\mathcal{E}$, to compute case 1 set of preferences $Prefs$.

- Line 3: Invoke Algorithm 3 with inputs $AAF$, $\mathcal{E}$ and $Prefs$, to compute case 2 preferences and combine them with case 1 preferences. This results in $PrefSet$ which is a set of sets of preferences.

- Line 4: Invoke Algorithm 4 with inputs $AAF$, $\mathcal{E}$ and $PrefSet$, to compute case 3 preferences and combine them with $Prefs$ and $PrefSet$. This results in an updated final $PrefSet$ which is a set of sets of preferences containing all three cases of preferences combined together.

- Line 5: Return the final $PrefSet$.

---

**Algorithm 1** Compute all preferences

---

**Require:** $AAF$, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict-free arguments
**Ensure:** $PrefSet$, the set of sets of all possible preferences
 1: **function** ComputeAllPreferences($AAF, \mathcal{E}$)
 2:     $Prefs \leftarrow$ ComputePreferences$_1$($AAF, \mathcal{E}$)
 3:     $PrefSet \leftarrow$ ComputePreferences$_2$($AAF, \mathcal{E}, Prefs$)
 4:     $PrefSet \leftarrow$ ComputePreferences$_3$($AAF, \mathcal{E}, PrefSet$)
 5:     **return** $PrefSet$
 6: **end function**

---

The following are the main steps in Algorithm 2:

- Line 3: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

- Line 4: Find all arguments $B$ that attack $A$.

- Lines $5 - 11$: For each $B$, if there is no unattacked argument $C$ (where $C \neq A$ and $C \in \mathcal{E}$) that attacks $B$, then compute each preference of the form $A > B$ and add it to the set of preferences $Prefs$.

The following are the main steps in Algorithm 3:

- Line 4: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

- Line 5: Find all arguments $B$ that $A$ attacks.

- Lines $6 - 13$: For all arguments $B$ attacked by $A$, compute preferences of the form $A > B$ and $A = B$, and add each preference relation to a different set of preferences, as per lines 8 and 9.

The following are the main steps in Algorithm 4:

- Line 3: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

- Line 4: Find all arguments $B$ that attack $A$.

- Lines $5 - 16$: For each $B$, if there is an unattacked argument $C$ (where $C \neq A$ and $C \in \mathcal{E}$) that attacks $B$, then compute preferences of the form $A > B$, $A = B$ and $B > A$, and add each preference relation to a different set of preferences, as per lines $9 - 11$.

We establish that our approach is sound (that is, all its outputs are correct) and complete (that is, it outputs all possible solutions). We start with its soundness:

---
**Algorithm 2** Compute preferences (Case 1)
---
**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict-free arguments
**Ensure:** *Prefs*, a set of preferences
  1: **function** ComputePreferences$_1$(*AAF*, $\mathcal{E}$)
  2:     $Prefs \leftarrow \emptyset$
  3:     **for all** $A \in \mathcal{E}$ **do**
  4:         $Attackers \leftarrow \{B \mid (B, A) \in \mathcal{R}\}$                          ▷ get all attackers of $A$
  5:         **for all** $B \in Attackers$ **do**
  6:             $Defenders \leftarrow \{C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}, \nexists X \in \mathcal{A}\ s.t.\ (X, C) \in \mathcal{R}\}$    ▷ $C$ attacks $B$ & defends $A$
  7:             **if** $Defenders = \emptyset$ **then**                          ▷ if $B$ not attacked by any $C$
  8:                 $Prefs \leftarrow Prefs \cup \{A > B\}$                      ▷ add preference $A > B$
  9:             **end if**
 10:         **end for**
 11:     **end for**
 12:     **return** *Prefs*
 13: **end function**
---

**Theorem 4.1.** *(Soundness): Algorithm 1 is sound in that given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, every output preference set Prefs $\in$ PrefSet, when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics).*

*Proof.* We prove this by exploring all cases and how these are handled by algorithms 2-4. Each set of preferences computed for each subset of arguments $\alpha, \beta, \gamma \subseteq \mathcal{A}$ is such that $\alpha, \gamma \subseteq \mathcal{E}, \beta \cap \mathcal{E} = \emptyset$. We proceed to show how each of the auxiliary algorithms 2-4 help us achieve this.

Algorithm 2 computing each case 1 preference of the form $A > B, A \in \mathcal{E}, B \in \beta, (B, A) \in \mathcal{R}$ ensures that the following holds:

1. There is no $C \in \mathcal{E}, C \neq A$ such that $(C, B) \in \mathcal{R}$ (lines 6-7).

2. $A \in \mathcal{E}$ since $A$ is preferred to its attacking argument $B$, which invalidates the attack $(B, A) \in \mathcal{R}$.

3. Since the input extension $\mathcal{E}$ consists of conflict free arguments, if $A \in \mathcal{E}$ then its attacking argument $B \notin \mathcal{E}$. This supports that $\beta \cap \mathcal{E} = \emptyset$.

Algorithm 3 computing each case 2 preferences of the form $A > B, A = B, A \in \mathcal{E}, B \in \beta, (A, B) \in \mathcal{R}, (B, A) \notin \mathcal{R}$ ensures the following holds:

1. Since $A$ attacks $B$ and $B$ does not attack $A$, we have two different preferences between $A$ and $B$, namely, $A > B, A = B$. Therefore $A \in \mathcal{E}$ with respect to each of these preferences.

2. Preferences $A > B, A = B$ will be in different preference sets, as per lines 8 and 9. We will have $Prefs_1 \leftarrow Prefs \cup \{A > B\}$ and $Prefs_2 \leftarrow Prefs \cup \{A = B\}$, where $Prefs$ consists of preferences of case 1.

Algorithm 4 computing each case 3 preferences of the form $A > B, A = B, B > A, A \in \alpha, B \in \beta, C \in \gamma, (B, A) \in \mathcal{R}, (C, B) \in \mathcal{R}$ ensures the following holds:

1. Since $C$ defends $A$ from the attack of $B$, we have three different preferences between $A$ and $B$, namely, $A > B$, $A = B$ and $B > A$. Therefore $A \in \mathcal{E}$ with respect to each of these preferences.

2. Preferences $A > B, A = B, B > A$ will be in different preference sets, as per lines 9, 10 and 11. We will have $Prefs_1 \leftarrow Prefs \cup \{A > B\}$, $Prefs_2 \leftarrow Prefs \cup \{A = B\}$ and $Prefs_3 \leftarrow Prefs \cup \{B > A\}$, where $Prefs$ consists of preferences of cases 1 and 2.

$\square$

**Algorithm 3** Compute preferences (Case 2)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict free arguments
**Require:** *Prefs*, a set of preferences
**Ensure:** *PrefSet*, a set of sets of preferences

1: **function** ComputePreferences$_2$(*AAF*, $\mathcal{E}$, *Prefs*)
2:     *PrefSet* $\leftarrow$ {*Prefs*}
3:     *PrefSet'* $\leftarrow \emptyset$
4:     **for all** $A \in \mathcal{E}$ **do**
5:         *Attacked* $\leftarrow \{B \mid (A, B) \in \mathcal{R} \wedge (B, A) \notin \mathcal{R}\}$                ▷ get arguments $A$ attacks
6:         **for all** $B \in$ *Attacked* **do**                        ▷ for all $B$ attacked by $A$
7:             **for all** *Prefs* $\in$ *PrefSet* **do**                 ▷ for all sets of preferences *Prefs*
8:                 *PrefSet'* $\leftarrow$ *PrefSet'* $\cup \{$*Prefs* $\cup \{A > B\}\}$             ▷ add *Prefs* $\cup \{A > B\}$
9:                 *PrefSet'* $\leftarrow$ *PrefSet'* $\cup \{$*Prefs* $\cup \{A = B\}\}$             ▷ add *Prefs* $\cup \{A = B\}$
10:             **end for**
11:             *PrefSet* $\leftarrow$ *PrefSet'*
12:             *PrefSet'* $\leftarrow \emptyset$
13:         **end for**
14:     **end for**
15:     **return** *PrefSet*
16: **end function**

---

**Theorem 4.2.** *(Completeness): Algorithm 1 is complete in that given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, if there is a preference set Prefs $\in$ PrefSet which when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics), then algorithm* 1 *will find it.*

*Proof.* Similar to above, we prove this by exploring all cases and how these are handled by algorithms 2-4. We find all sets of preferences computed for each subset of arguments $\alpha, \beta, \gamma \subseteq \mathcal{A}, \alpha, \gamma \subseteq \mathcal{E}, \beta \cap \mathcal{E} = \emptyset$. We proceed to show how each of the auxiliary algorithms 2-4 help us achieve this.

Algorithm 2 computes all case 1 preferences of the form $A > B, A \in \mathcal{E}, B \in \beta, (B, A) \in \mathcal{R}$. Lines 3-11 exhaustively search for $A \in \mathcal{E}$ for which there is an attacker $B$ (not attacked by any $C \neq A$). If there are such $A, B \in \mathcal{A}$, the algorithm will find them and add $A > B$ to a set of preferences.

Algorithm 3 computes all case 2 preferences of the form $A > B, A = B, B \in \beta, A \in \mathcal{E}, (A, B) \in \mathcal{R}, (B, A) \notin \mathcal{R}$. Lines 4-14 exhaustively search for $A \in \mathcal{E}$ for which there is an attacked argument $B$ and $B$ does not attack $A$. If there are such $A, B \in \mathcal{A}$, the algorithm will find them and add each $A > B, A = B$ to a different set of preferences.

Algorithm 4 computes all case 3 preferences of the form $A > B, A = B, B > A, A \in \alpha, B \in \beta, C \in \gamma, (B, A) \in \mathcal{R}, (C, B) \in \mathcal{R}$. Lines 3-17 exhaustively search for $A \in \mathcal{E}$ for which there is an attacker $B$ and there is a defender $C$ that attacks $B$. If there are such $A, B, C \in \mathcal{A}$, the algorithm will find them and add each $A > B, A = B, B > A$ to a different set of preferences. $\square$

After having proved the soundness and completeness of Algorithm 1, we establish its termination.

**Theorem 4.3.** *(Termination): Given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, Algorithm 1 always terminates.*

*Proof.* Algorithm 1 invokes Algorithms 2, 3 and 4 in lines 2-4 to compute a set of all sets of preferences. To prove Algorithm 1 terminates we consider the termination of Algorithms 2, 3 and 4 individually. Since we assume that both the input abstract argumentation framework *AAF* and an extension $\mathcal{E}$ are finite, therefore the *for loop* which iterates over all the elements of the extension in Algorithms 2, 3 and 4 will always terminate. The rest of the proof explores each algorithm in turn.

In Algorithm 2, since the set of *Attackers*, i.e., all the arguments that attack $A$, is finite therefore the *for loop* in lines 5-10 will always terminate. In Algorithm 3, since the set of *Attacked*, i.e., all the arguments that are attacked by

---

**Algorithm 4** Compute preferences (Case 3)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict free arguments
**Require:** *PrefSet*, a set of sets of preferences
**Ensure:** *PrefSet*, an updated set of sets of preferences

1:  **function** ComputePreferences₃(*AAF*, $\mathcal{E}$, *PrefSet*)
2:      *PrefSet′* ← ∅
3:      **for all** $A \in \mathcal{E}$ **do**
4:          *Attackers* ← {$B \mid (B, A) \in \mathcal{R}$}                                    ▷ get all attackers of $A$
5:          **for all** $B \in$ *Attackers* **do**
6:              *Defenders* ← {$C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}, \nexists X \in \mathcal{A}$ *s.t.* $(X, C) \in \mathcal{R}$}        ▷ $C$ attacks $B$ & defends $A$
7:              **if** *Defenders* ≠ ∅ **then**
8:                  **for all** *Prefs* $\in$ *PrefSet* **do**                        ▷ for all sets of preferences *Prefs*
9:                      *PrefSet′* ← *PrefSet′* ∪ {*Prefs* ∪ {$A > B$}}                        ▷ add *Prefs* ∪ {$A > B$}
10:                     *PrefSet′* ← *PrefSet′* ∪ {*Prefs* ∪ {$A = B$}}                        ▷ add *Prefs* ∪ {$A = B$}
11:                     *PrefSet′* ← *PrefSet′* ∪ {*Prefs* ∪ {$B > A$}}                        ▷ add *Prefs* ∪ {$B > A$}
12:                 **end for**
13:                 *PrefSet* ← *PrefSet′*
14:                 *PrefSet′* ← ∅
15:             **end if**
16:         **end for**
17:     **end for**
18:     **return** *PrefSet*
19: **end function**

---

$A$, is finite therefore the *for loop* in lines 6-13 will always terminate. Furthermore, since the set of sets of preferences *PrefSet* is finite, therefore the *for loop* in lines 7-10 will always terminate. In Algorithm 4, since the set of *Attackers*, i.e., all the arguments that attack $A$ is finite, therefore the *for loop* in lines 5-16 will always terminate. Furthermore, since the set of sets of preferences *PrefSet* is finite, so the *for loop* in lines 8-12 will always terminate.

Thus, we have proved that Algorithms 2, 3 and 4 terminate. Therefore, Algorithm 1 that invokes Algorithms 2, 3 and 4 always terminates.                                                                                                            □

### 4.1.1. Algorithms for Filtering Preferences

Additionally, we present Algorithm 5 to compute the unique preferences for an extension in comparison to another extension, and Algorithm 6 to compute the common preferences for any two extensions. Algorithm 5 takes as input two different set of of sets of preferences *PrefSet₁* and *PrefSet₂* for two different extensions, and computes the set of unique preferences that do not overlap between *PrefSet₁* and *PrefSet₂*. The following are the main steps in Algorithm 5:

- Line 2: Iterate over each element *Prefs₁* in the set of sets of preferences *PrefSet₁*.

- Line 3: Iterate over each element $p$ in the set of preferences *Prefs₁*.

- Lines 4-6: Check the condition if there is no set of preferences *Prefs₂* in the set of sets of preferences *PrefSet₂*, which consists of $p$, then add $p$ to the unique set of preferences *UniquePrefs*.

- Line 5: Return the unique set of preferences *UniquePrefs*.

Algorithm 6 takes as input two different set of sets of preferences *PrefSet₁* and *PrefSet₂* for two different extensions, and computes the set of common preferences that overlap between *PrefSet₁* and *PrefSet₂*. The following are the main steps in Algorithm 6:

- Line 2: Iterate over each element *Prefs₁* in the set of sets of preferences *PrefSet₁*.

---

**Algorithm 5** Algorithm for Computing Unique Preferences

---

**Require:** *PrefSet₁*, the set of sets of preferences for first extension.
**Require:** *PrefSet₂*, the set of sets of preferences for second extension.
**Ensure:** *UniquePrefs*, unique preferences for first extension.
 1: **function** ComputeUniquePreferences(*PrefSet1*, *PrefSet2*)
 2:    **for all** $Prefs_1 \in PrefSet_1$ **do**
 3:       **for all** $p \in Prefs_1$ **do**
 4:          **if** $\nexists Prefs_2 \in PrefSet_2$ *s.t.* $p \in Prefs_2$ **then**
 5:             $UniquePrefs \leftarrow UniquePrefs \cup p$
 6:          **end if**
 7:       **end for**
 8:    **end for**
 9:    **return** *UniquePrefs*
10: **end function**

---

- Line 3: Iterate over each element $p$ in the set of preferences *Prefs₁*.

- Lines 4-6: Check the condition if there is a set of preferences *Prefs₂* in the set of sets of preferences *PrefSet₂*, which consists of $p$, then add $p$ to the common set of preferences *CommonPrefs*.

- Line 5: Return the common set of preferences *CommonPrefs*.

---

**Algorithm 6** Algorithm for Computing Common Preferences

---

**Require:** *PrefSet₁*, the set of sets of preferences for first extension.
**Require:** *PrefSet₂*, the set of sets of preferences for second extension.
**Ensure:** *CommonPrefs*, common preferences for both extensions.
 1: **function** ComputeCommonPreferences(*PrefSet₁*, *PrefSet₂*)
 2:    **for all** $Prefs_1 \in PrefSet_1$ **do**
 3:       **for all** $p \in Prefs_1$ **do**
 4:          **if** $\exists Prefs_2 \in PrefSet_2$ *s.t.* $p \in Prefs_2$ **then**
 5:             $CommonPrefs \leftarrow CommonPrefs \cup p$
 6:          **end if**
 7:       **end for**
 8:    **end for**
 9:    **return** *CommonPrefs*
10: **end function**

---

*4.1.2. Illustrative Example*

In this section, we present an illustrative example to demonstrate how Algorithm 1 works. Suppose we have an input abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ shown in Figure 1, where $\mathcal{A} = \{A, B, C, D, E\}$ and $\mathcal{R} = \{(A, B), (C, B), (C, D), (D, C), (D, E)\}$. We consider the conflict-free extension $\mathcal{E}_1 = \{A, C, E\}$ for computing preferences. Table 1 shows the preferences computed in lines 2, 3 and 4 of Algorithm 1.

- On line 2, Algorithm 2 is invoked, which returns the set of case 1 preferences $\{C > D\}$.

- On line 3, Algorithm 3 is invoked, which returns a set of sets of preferences (cases 1 and 2 combined together) $\{\{C > D, A > B, C > B\}, \{C > D, A > B, C = B\}, \{C > D, A = B, C > B\}, \{C > D, A = B, C = B\}\}$.

- Finally on line 4, Algorithm 4 is invoked, which returns a set of sets of preferences (cases 1, 2 and 3 combined together) $\{\{C > D, A > B, C > B, E > D\}, \{C > D, A > B, C > B, E = D\}, \{C > D, A > B, C > B, D > E\}, \{C > D, A > B, C = B, E > D\}, \{C > D, A > B, C = B, E = D\}, \{C > D, A > B, C = B, D > E\},$

Table 1: Computing Preferences for Extension $\{A, C, E\}$

| Line No. | Preference Sets |
|---|---|
| 2 | $\{C > D\}$ |
| 3 | $\{\{C > D, A > B, C > B\},$<br>$\{C > D, A > B, C = B\},$<br>$\{C > D, A = B, C > B\},$<br>$\{C > D, A = B, C = B\}\}$ |
| 4 | $\{\{C > D, A > B, C > B, E > D\},$<br>$\{C > D, A > B, C > B, E = D\},$<br>$\{C > D, A > B, C > B, D > E\},$<br>$\{C > D, A > B, C = B, E > D\},$<br>$\{C > D, A > B, C = B, E = D\},$<br>$\{C > D, A > B, C = B, D > E\},$<br>$\{C > D, A = B, C > B, E > D\},$<br>$\{C > D, A = B, C > B, E = D\},$<br>$\{C > D, A = B, C > B, D > E\},$<br>$\{C > D, A = B, C = B, E > D\},$<br>$\{C > D, A = B, C = B, E = D\},$<br>$\{C > D, A = B, C = B, D > E\}\}$ |

$\{C > D, A = B, C > B, E > D\}, \{C > D, A = B, C > B, E = D\}, \{C > D, A = B, C > B, D > E\},$
$\{C > D, A = B, C = B, E > D\}, \{C > D, A = B, C = B, E = D\}, \{C > D, A = B, C = B, D > E\}\}$.

Table 2 presents the sets of preferences for the two preferred extensions $\{A, C, E\}$ and $\{A, D\}$ of the abstract argumentation framework given above and shown in Figure 1. The sets of preferences for all conflict-free extensions for this example abstract argumentation framework are shown in Table A.3 in the Appendix.

Table 2: Preferences for the Preferred extensions $\{A, C, E\}$ and $\{A, D\}$

| Preferred Extensions | Preference Sets | Unique Preferences | Common Preferences |
|---|---|---|---|
| $\{A, C, E\}$ | $\{\{C > D, A > B, C > B, E > D\},$<br>$\{C > D, A > B, C > B, E = D\},$<br>$\{C > D, A > B, C > B, D > E\},$<br>$\{C > D, A > B, C = B, E > D\},$<br>$\{C > D, A > B, C = B, E = D\},$<br>$\{C > D, A > B, C = B, D > E\},$<br>$\{C > D, A = B, C > B, E > D\},$<br>$\{C > D, A = B, C > B, E = D\},$<br>$\{C > D, A = B, C > B, D > E\},$<br>$\{C > D, A = B, C = B, E > D\},$<br>$\{C > D, A = B, C = B, E = D\},$<br>$\{C > D, A = B, C = B, D > E\}\}$ | $C > D$<br>$E > D$<br>$C > B$<br>$C = B$ | $A > B$<br>$A = B$<br>$D > E$<br>$D = E$ |
| $\{A, D\}$ | $\{\{D > C, A > B, D > E\},$<br>$\{D > C, A > B, D = E\},$<br>$\{D > C, A = B, D > E\},$<br>$\{D > C, A = B, D = E\}\}$ | $D > C$ | |

The unique preferences for an extension in comparison to another extension can be computed by Algorithm 5. By analysing the preference sets shown in Table 2, we can identify the unique preferences for extension $\{A, C, E\}$, which

are $C > D$, $E > D$, $C > B$ and $C = B$[4], and the unique preferences for extension $\{A, D\}$, which is $D > C$. Since, at least one unique preference for each extension is in its corresponding preference set, therefore it can be concluded that if we evaluate the example abstract argumentation framework given in Figure 1 with a corresponding preference set of a given preferred extension, then the evaluation results in exactly the same preferred extension.

Furthermore, we can identify preferences that are common to both extensions, which are $A > B$, $A = B$, $D > E$, $D = E$[5]. The common preferences for any two extensions can be computed by Algorithm 6. It is interesting to note that, extension $\{A, C, E\}$ can have preferences $D > E$ and $D = E$, considering $D$ is not present in the extension. It can be concluded that if we evaluate the example abstract argumentation framework given in Figure 1, then we get both preferred extensions with the following preference sets: $\{A > B, D > E\}$, $\{A > B, D = E\}$, $\{A = B, D > E\}$ and $\{A = B, D = E\}$.

### 4.2. An Approximate Algorithm for Computing Preferences

While Algorithm 1 can compute a set of all sets of preferences using the three cases described earlier, the number of the possible sets of preferences increases exponentially with regards to the number of arguments within the abstract argumentation framework, resulting in exponential time complexity. This is impractical for a large set of arguments, and in this section, we describe an approximate method for computing a set of preferences.

We now present and describe Algorithm 7 that approximately computes a possible set of preferences for a given input extension (consisting of conflict-free arguments) in an abstract argumentation framework (AAF) using the three cases described earlier. The input of Algorithm 7 is a tuple $\langle AAF, \mathcal{E} \rangle$, where:

- Abstract argumentation framework $AAF = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{A}$ denotes the set of all arguments in the $AAF$, and $\mathcal{R}$ denotes the attack relation between arguments.

- Extension $\mathcal{E}$ consists of a finite number of conflict-free arguments such that $\mathcal{E} \subseteq \mathcal{A}$.

The algorithm computes and outputs a finite set of preferences, which is represented as $Prefs = \{A > B, C > B, ....\}$ such that $\{A, B, C, ...\} \subseteq \mathcal{A}$.

The following are the main steps in Algorithm 7:

- Lines $3 - 11$: Compute all Case 1 preferences.

  - Line 3: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

  - Line 4: Find all arguments $B$ that attack $A$.

  - Lines $5 - 10$: For each $B$, if there is no unattacked argument $C$ (where $C \neq A$ and $C \in \mathcal{E}$) that attacks $B$, then compute each preference of the form $A > B$ and add it to the set of preferences $Prefs$.

- Lines $12 - 18$: Compute Case 2 preferences and combine them with Case 1 preferences.

  - Line 12: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

  - Line 13: Find all argument $B$ that $A$ attacks.

  - Lines $14 - 17$: For all arguments $B$ attacked by $A$, generate a random preference $p$ such that $p \in \{A > B, A = B\}$, and add it to the set of preferences $Prefs$, as per lines $15 - 16$.

- Lines $19 - 28$: Compute Case 3 preferences and combine them with Case 2 and Case 3 preferences.

  - Line 19: Iteratively pick a single argument $A$ from the extension $\mathcal{E}$.

  - Line 20: Find all arguments $B$ that attack $A$.

  - Lines $21 - 27$: For each $B$, if there is an unattacked argument $C$ (where $C \neq A$ and $C \in \mathcal{E}$) that attacks $B$, then generate a random preference $p$ such that $p \in \{A > B, A = B, B > A\}$, and add it to the set of preferences $Prefs$, as per lines $24 - 25$.

---

[4]This means it could be either $C > B$ or $C = B$.
[5]This means it could be either $A > B$ or $A = B$, and similarly $D > E$ or $D = E$.

**Algorithm 7** Approximately compute preferences

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict-free arguments
**Ensure:** *Prefs*, an approximate set of possible preferences

1:  **function** ComputePreferencesApproximately(*AAF*, $\mathcal{E}$)
2:      *Prefs* $\leftarrow \emptyset$
3:      **for all** $A \in \mathcal{E}$ **do**
4:          *Attackers* $\leftarrow \{B \mid (B, A) \in \mathcal{R}\}$                                                  ▷ get all attackers of *A*
5:          **for all** $B \in$ *Attackers* **do**
6:              *Defenders* $\leftarrow \{C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}, \nexists X \in \mathcal{A} \ s.t. \ (X, C) \in \mathcal{R}\}$   ▷ *C* attacks *B* & defends *A*
7:              **if** *Defenders* $= \emptyset$ **then**                                               ▷ if *B* not attacked by any *C*
8:                  *Prefs* $\leftarrow$ *Prefs* $\cup \{A > B\}$                                    ▷ add preference *A > B*
9:              **end if**
10:         **end for**
11:     **end for**
12:     **for all** $A \in \mathcal{E}$ **do**
13:         *Attacked* $\leftarrow \{B \mid (A, B) \in \mathcal{R} \wedge (B, A) \notin \mathcal{R}\}$                          ▷ get arguments *A* attacks
14:         **for all** $B \in$ *Attacked* **do**                                                  ▷ for all *B* attacked by *A*
15:             Generate a random preference $p$ such that $p \in \{A > B, \ A = B\}$
16:             *Prefs* $\leftarrow$ *Prefs* $\cup \ p$                                                   ▷ add preference *p*
17:         **end for**
18:     **end for**
19:     **for all** $A \in \mathcal{E}$ **do**
20:         *Attackers* $\leftarrow \{B \mid (B, A) \in \mathcal{R}\}$                                                 ▷ get all attackers of *A*
21:         **for all** $B \in$ *Attackers* **do**
22:             *Defenders* $\leftarrow \{C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}, \nexists X \in \mathcal{A} \ s.t. \ (X, C) \in \mathcal{R}\}$   ▷ *C* attacks *B* & defends *A*
23:             **if** *Defenders* $\neq \emptyset$ **then**                                               ▷ if *B* is attacked by any *C*
24:                 Generate a random preference $p$ such that $p \in \{A > B, \ A = B, \ B > A\}$
25:                 *Prefs* $\leftarrow$ *Prefs* $\cup \ p$                                                ▷ add preference *p*
26:             **end if**
27:         **end for**
28:     **end for**
29:     **return** *Prefs*
30: **end function**

---

- Line 29: Return the final set of preferences *Prefs*.

    We establish that our approach is sound (that is, its output is correct).

**Theorem 4.4.** *(Soundness): Algorithm 7 is sound in that given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, the output preference set Prefs, when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics).*

*Proof.* We prove this by exploring all cases and how these are handled by the algorithm. The set of preferences computed for each subset of arguments $\alpha, \beta, \gamma \subseteq \mathcal{A}$ is such that $\alpha, \gamma \subseteq \mathcal{E}, \beta \cap \mathcal{E} = \emptyset$.
Lines $3 - 11$ computing each case 1 preference of the form $A > B, A \in \mathcal{E}, B \in \beta, (B, A) \in \mathcal{R}$ ensure that the following holds:

1. There is no $C \in \mathcal{E}, C \neq A$ such that $(C, B) \in \mathcal{R}$.

2. $A \in \mathcal{E}$ since $A$ is preferred to its attacking argument $B$, which invalidates the attack $(B, A) \in \mathcal{R}$.

3. Since the input extension $\mathcal{E}$ consists of conflict free arguments, if $A \in \mathcal{E}$ then its attacking argument $B \notin \mathcal{E}$. This supports that $\beta \cap \mathcal{E} = \emptyset$.

Lines $12 - 18$ computing a random case 2 preference $p$ of the form $p \in \{A > B, A = B\}, A \in \mathcal{E}, B \in \beta, (A, B) \in \mathcal{R}, (B, A) \notin \mathcal{R}$ ensure the following holds:

1. Since $A$ attacks $B$ and $B$ does not attack $A$, we have two different preferences between $A$ and $B$, namely, $A > B, A = B$. Therefore $A \in \mathcal{E}$ with respect to each of these preferences.

2. A randomly generated preference $p \in \{A > B, A = B\}$ will be added to the preference set *Prefs*, as per lines 15 and 16. We will have *Prefs* $\leftarrow$ *Prefs* $\cup p$, where *Prefs* consists of preferences of case 1.

Lines $19 - 28$ computing a random case 3 preference of the form $p \in \{A > B, A = B, B > A\}, A \in \alpha, B \in \beta, C \in \gamma, (B, A) \in \mathcal{R}, (C, B) \in \mathcal{R}$ ensure the following holds:

1. Since $C$ defends $A$ from the attack of $B$, we have three different preferences between $A$ and $B$, namely, $A > B$, $A = B$ and $B > A$. Therefore $A \in \mathcal{E}$ with respect to each of these preferences.

2. A randomly generated preference $p \in \{A > B, A = B, B > A\}$ will be added to the preference set *Prefs*, as per lines 24 and 25. We will have *Prefs* $\leftarrow$ *Prefs* $\cup p$, where *Prefs* consists of preferences of cases 1 and 2.

$\square$

After having proved the soundness of Algorithm 7, we establish its termination.

**Theorem 4.5.** *(Termination): Given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, Algorithm 7 always terminates.*

*Proof.* Algorithm 7 consists of three *for loops* for computing each of the case 1 (in lines $3 - 11$), case 2 (in lines $12 - 18$), and case 3 (in lines $19 - 28$) preferences for the output set of preferences. To prove Algorithm 7 terminates we consider the termination of each of the *for loops* for the three cases individually. Since we assume that both the input abstract argumentation framework *AAF* and an extension $\mathcal{E}$ are finite, therefore the three *for loops* which iterate over all the elements of the extension in lines $3 - 11$, lines $12 - 18$ and lines $19 - 28$ will always terminate. The rest of the proof explores each case in turn.

Within the *for loop* in lines $3 - 11$, since the set of *Attackers*, i.e., all the arguments that attack $A$, is finite therefore the *for loop* in lines 5-10 will always terminate. Within the *for loop* in lines $12 - 18$, since the set of *Attacked*, i.e., all the arguments that are attacked by $A$, is finite therefore the *for loop* in lines 14-17 will always terminate. Within the *for loop* in lines $19 - 28$, since the set of *Attackers*, i.e., all the arguments that attack $A$ is finite, therefore the *for loop* in lines 21-27 will always terminate.

Thus, we have proved that Algorithm 7 always terminates. $\square$

## 5. Verifying Preferences

As mentioned in Section 2, preference-based argumentation frameworks [20] ignore or remove the attacks where the attacked argument is stronger than the attacking argument. It was found out later that the resulting extension violates the basic condition imposed on acceptability semantics, which is the conflict-freeness of extensions. This problem was later resolved in a new preference-based argumentation framework that guarantees conflict-free extensions with a symmetric conflict relation [29, 24, 21]. The preference relation is then used to determine the direction of the defeat relation between the two arguments.

Since we assume our input extension to be conflict-free both approaches of attack removal or reversal would work. Therefore, we consider two methods for the application of preferences, and verifying that this results in our desired input extension. A preference-based argumentation framework (PAF) can be transformed into an abstract argumentation framework (AAF) by:

1. applying preferences by attack removal, or,

2. applying preferences by attack reversal.

We now present the formal definitions for both methods of applying preferences to transform a PAF into an AAF.

Figure 3: Transformed abstract argumentation framework $AAF_3$

**Definition 5.1.** *(Applying preferences by attack removal): A preference-based argumentation framework $(\mathcal{A}, Def, \geq)$ can be transformed into an abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ as follows: $\forall A, B \in \mathcal{A}$, it is the case that $(B, A) \in \mathcal{R}$ iff $(B, A) \in Def$ and it is not the case that $A > B$.*

In other words, if $B$ defeats $A$ and $A$ is preferred to $B$ then the attack $(B, A)$ will not appear in the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$.

**Definition 5.2.** *(Applying preferences by attack reversal): A preference-based argumentation framework $(\mathcal{A}, Def, \geq)$ can be transformed into an abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ as follows:*

  *(i)  $\forall A, B \in \mathcal{A}$, it is the case that $(B, A) \notin \mathcal{R}$ and $(A, B) \in \mathcal{R}$ iff $(B, A) \in Def$ and it is the case that $A > B$.*

  *(ii)  $\forall A, B \in \mathcal{A}$, it is the case that $(B, A) \in \mathcal{R}$ iff $(B, A) \in Def$ and it is not the case that $A > B$.*

In other words, if $B$ defeats $A$ and $A$ is preferred to $B$ then the attack $(B, A)$ will not appear in the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$, and the reverse attack $(A, B)$ will appear in the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$. Also, if $B$ defeats $A$ and $A$ is not preferred to $B$ then the attack $(B, A)$ will appear in the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$.

**Example 5.1.** *We refer back to the abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ of Figure 1, where $\mathcal{A} = \{A, B, C, D, E\}$ and $\mathcal{R} = \{(A, B), (C, B), (C, D), (D, C), (D, E)\}$. We consider the preferred extension $\mathcal{E}_1 = \{A, C, E\}$ to compute the following set of sets of preferences using Algorithm 1:*

$$\{\{C > D, A > B, C > B, E > D\},$$
$$\{C > D, A > B, C > B, E = D\},$$
$$\{C > D, A > B, C > B, D > E\},$$
$$\{C > D, A > B, C = B, E > D\},$$
$$\{C > D, A > B, C = B, E = D\},$$
$$\{C > D, A > B, C = B, D > E\},$$
$$\{C > D, A = B, C > B, E > D\},$$
$$\{C > D, A = B, C > B, E = D\},$$
$$\{C > D, A = B, C > B, D > E\},$$
$$\{C > D, A = B, C = B, E > D\},$$
$$\{C > D, A = B, C = B, E = D\},$$
$$\{C > D, A = B, C = B, D > E\},\}$$

*We consider the first set of preferences $Pref_1 = \{C > D, A > B, C > B, E > D\}$ to demonstrate the application of preferences. The corresponding preference-based argumentation framework is denoted as $PAF = (\mathcal{A}, Def, \geq)$ which represents the AAF of Figure 1 with a set of preferences $Pref_1 = \{C > D, A > B, C > B, E > D\}$. The preferences can be applied using two different methods, attack removal and attack reversal, defined earlier in Definitions 5.1 and 5.2 to transform the PAF to an AAF. The application of the first method, i.e., attack removal, results in an abstract argumentation framework shown in Figure 3, where the attacks $(D, C) \in \mathcal{R}$ and $(D, E) \in \mathcal{R}$ have been removed. The application of the second method, i.e., attack reversal, results in an abstract argumentation framework shown in Figure 4, where the attack $(C, D) \in \mathcal{R}$ represented as a dashed arrow denotes both the normal and reverse attacks and the attack $(E, D) \in \mathcal{R}$ represented as a dotted arrow denotes reverse attack.*

*To verify that the set of preferences $Pref_1 = \{C > D, A > B, C > B, E > D\}$ is correct. We apply the preferences to get the transformed AAFs shown in Figures 3 and 4. The resulting preferred extension for both transformed AAFs is $\{A, C, E\}$. Thus, the set of preferences $Pref_1$ has been verified to be correct.*
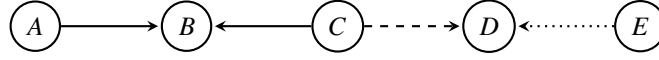
Figure 4: Transformed abstract argumentation framework $AAF_4$

### 5.1. Algorithms for Verifying Preferences

In order to assess and verify whether the computed set of preferences over arguments is correct, we need to test that the set of preferences when applied to a given abstract argumentation framework results in our original input extension under a given semantics. We now state the verification problem precisely as follows.

**Problem 5.1.** *Given an abstraction argumentation framework AAF, an extension $\mathcal{E}$, a semantics $\sigma \in \{grounded, preferred, stable\}$, and a set of sets of preferences PrefSet, each preference set Prefs $\in$ PrefSet when applied to the AAF results in the single input extension $\mathcal{E}$ under a given semantics $\sigma$.*

To solve Problem 5.1, we present and describe Algorithms 8 and 10 for the verification of preferences following the two methods described above[6]. The input to both algorithms is a tuple $\langle AAF, \mathcal{E}, \sigma, PrefSet \rangle$, consisting of:

- An abstract argumentation framework $AAF = (\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$).

- An extension $\mathcal{E}$ consists of a finite number of conflict-free arguments such that $\mathcal{E} \subseteq \mathcal{A}$.

- $\sigma$ is a given semantic, where $\sigma \in \{grounded, preferred, stable\}$.

- *PrefSet* is a set of sets of preferences, where each set of preferences *Prefs* $\in$ *PrefSet* is represented as *Prefs* = $\{A > B, B = C, ....\}$ such that $\{A, B, C, ....\} \subseteq \mathcal{A}$.

Both algorithms output a Boolean variable *vcheck*, which is *true* if all *Prefs* $\in$ *PrefSet* are correct, otherwise it is *false*.

The main functionalities of Algorithm 8 are:

- *ApplyPreferences1* applies the preferences using the first method of attack removal in the abstract argumentation framework given in Definition 5.1.

- *ComputeExtensions* computes and returns all the extensions in the abstract argumentation framework given a particular semantics $\sigma$[7].

- The set of preferences are verified by getting the transformed abstract argumentation framework returned by *ApplyPreferences1* and checking that the resulting extension for a given semantics $\sigma$ returned by *ComputeExtensions* function is the same (and only) as the input extension.

---

[6]Please note, although an algorithm for preference application by attack removal will suffice for this work, we have presented the algorithm for preference application by attack reversal for comparison of both approaches in our experiments that are presented later.

[7]We note that this functionality is not defined as we use an existing function implemented in the Tweety library [40] for this purpose.

**Algorithm 8** Verify Preferences (Attack Removal)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict-free arguments
**Require:** $\sigma$, semantics for computing the extension
**Require:** *PrefSet*, a set of sets of preferences
**Ensure:** *vcheck*, a boolean value indicating preference verification success or failure

 1: **function** VerifyPreferences$_1$($AAF, \mathcal{E}, \sigma, PrefSet$)
 2:　　　$count \leftarrow 0$　　　　　　　　　　　　　　　　▷ number of correct sets of preferences
 3:　　　**for all** *Prefs* $\in$ *PrefSet* **do**
 4:　　　　　$AAF' \leftarrow$ ApplyPreferences$_1$($AAF, Prefs$)
 5:　　　　　$\mathcal{E}_s \leftarrow$ ComputeExtensions($AAF', \sigma$)
 6:　　　　　**if** $|\mathcal{E}_s| = 1$ & $\mathcal{E}' \in \mathcal{E}_s$ s.t. $\mathcal{E}' = \mathcal{E}$ **then**　　　▷ computed extension is equal to $\mathcal{E}$
 7:　　　　　　　$count \leftarrow count + 1$　　　　　　　　　　　　　　　▷ increment *count*
 8:　　　　　**end if**
 9:　　　**end for**
10:　　　**if** $|PrefSet| = |count|$ **then**　　　　　▷ all sets of preferences in *PrefSet* are correct
11:　　　　　$vcheck \leftarrow true$
12:　　　**else**
13:　　　　　$vcheck \leftarrow false$
14:　　　**end if**
15:　　　**return** *vcheck*
16: **end function**

---

**Algorithm 9** Apply Preferences (Attack Removal)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** *Prefs*, a set of preferences
**Ensure:** $AAF'$, an updated abstract argumentation framework

 1: **function** ApplyPreferences$_1$($AAF, Prefs$)
 2:　　　**for all** $(B, A) \in \mathcal{R}$ **do**
 3:　　　　　**if** $A > B \notin Prefs$ **then**　　　　　　　▷ $A$ is not preferred to its attacker $B$
 4:　　　　　　　$\mathcal{R}' \leftarrow \mathcal{R}' \cup \{(B, A)\}$　　　　　　　　　　　　　　▷ add attack
 5:　　　　　**end if**
 6:　　　**end for**
 7:　　　$AAF' \leftarrow (\mathcal{A}, \mathcal{R}')$　　　　　▷ argumentation framework $AAF'$ with attack relation $\mathcal{R}'$
 8:　　　**return** $AAF'$
 9: **end function**

---

We establish that Algorithm 8 is sound (that is, its output is correct) and complete (that is, it outputs all solutions). We start with its soundness:

**Theorem 5.1.** *(Soundness): Algorithm 8 is sound in that given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets of preferences PrefSet, and semantics $\sigma$ as input, each preference set Prefs $\in$ PrefSet, when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics $\sigma$).*

*Proof.* We prove this by first proving the auxiliary Algorithm 9 for applying preferences. Algorithm 9 goes through all attacks $(B, A) \in \mathcal{R}$ in the input AAF and adds the attack $(B, A)$ to the attack relation $\mathcal{R}'$ where the attacked argument $A$ is not preferred to the attacking argument $B$ as per lines $3 - 4$. This ensures that any attacks of the form $(B, A)$ where the attacked argument $A$ is preferred to the attacking argument $B$ are not added to the attack relation $\mathcal{R}'$. This results in the transformed $AAF' = (\mathcal{A}, \mathcal{R}')$.

Verification is then performed by invoking *ComputeExtensions* that computes and returns all the extensions in the transformed abstract argumentation framework $AAF'$ for a given semantics $\sigma$. Since, if the output is a single extension

which is equal to the input extension as per lines $6 - 8$, therefore *Prefs* is verified to be correct. Thus, *PrefSet* is then verified to be correct if the verification for each preference set *Prefs* $\in$ *PrefSet* is correct as per lines $10 - 14$.

$\square$

**Theorem 5.2.** *(Completeness): Algorithm 8 is complete in that given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets preferences PrefSet, and semantics $\sigma$ as input, each Prefs $\in$ PrefSet is verified to be correct, i.e., each Prefs $\in$ PrefSet when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics $\sigma$).*

*Proof.* Algorithm 8 goes through each *Prefs* $\in$ *PrefSet* to verify its correctness, as per lines $3 - 9$. It then invokes Algorithm 9 to apply each *Prefs* to the *AAF* and get the updated *AAF'* as per line 4. *ComputeExtensions* is then invoked to get the extension of the updated *AAF'* under a given semantics $\sigma$.

Algorithm 9 goes through all attacks $(B, A) \in \mathcal{R}$ in the input AAF and adds the attack $(B, A)$ to the attack relation $\mathcal{R}'$ where the attacked argument $A$ is not preferred to the attacking argument $B$ as per lines $2 - 6$.

Lines $10 - 14$ in Algorithm 8 ensure that all *Prefs* $\in$ *PrefSet* are verified to be correct. $\square$

After having proved the soundness and completeness of Algorithm 8, we establish its termination.

**Theorem 5.3.** *(Termination): Given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets preferences PrefSet, and semantics $\sigma$ as input, Algorithm 8 always terminates.*

*Proof.* Algorithm 8 consists of one *for loop* that goes through each *Prefs* $\in$ *PrefSet* to verify its correctness, as per lines $3 - 9$. Since we assume that the input *PrefSet* is finite, therefore the *for loop* which iterate over all the elements *Prefs* $\in$ *PrefSet* in lines $3 - 9$ will always terminate.

Algorithm 8 invokes Algorithm 9 to apply each *Prefs* $\in$ *PrefSet* to the *AAF* and get the updated *AAF'*. We now prove that Algorithm 9 terminates. Since we assume that the attack relation $\mathcal{R}$ of the input *AAF* is finite, therefore the *for loop* which iterates over all the elements $(B, A) \in \mathcal{R}$ in lines $2 - 6$ will always terminate.

Thus, we have proved that Algorithm 8 always terminates. $\square$

The main functionalities of Algorithm 10 are:

- *ApplyPreferences2* applies the preferences using the second method of attack reversal in the abstract argumentation framework given in Definition 5.2.

- *ComputeExtensions* computes and returns all the extensions in the abstract argumentation framework given a particular semantics $\sigma$.

- The set of preferences are verified by getting the transformed abstract argumentation framework returned by *ApplyPreferences2* and checking that the resulting extension for a given semantic $\sigma$ returned by *ComputeExtensions* function is the same (and only) as the input extension.

**Algorithm 10** Verify Preferences (Attack Reversal)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** $\mathcal{E}$, an extension consisting of conflict-free arguments
**Require:** $\sigma$, semantics for computing the extension
**Require:** *PrefSet*, a set of sets of preferences
**Ensure:** *vcheck*, a boolean value indicating preference verification success or failure

```
 1: function VerifyPreferences₂(AAF, 𝓔, σ, PrefSet)
 2:     count ← 0                                          ▷ number of correct sets of preferences
 3:     for all Prefs ∈ PrefSet do
 4:         AAF′ ← ApplyPreferences₂(AAF, Prefs)
 5:         𝓔ₛ ← ComputeExtensions(AAF′, σ)
 6:         if |𝓔ₛ| = 1 & 𝓔′ ∈ 𝓔ₛ s.t. 𝓔′ = 𝓔 then        ▷ computed extension is equal to 𝓔
 7:             count ← count + 1                                              ▷ increment count
 8:         end if
 9:     end for
10:     if |PrefSet| = |count| then                ▷ all sets of preferences in PrefSet are correct
11:         vcheck ← true
12:     else
13:         vcheck ← false
14:     end if
15:     return vcheck
16: end function
```

---

**Algorithm 11** Apply Preferences (Attack Reversal)

---

**Require:** *AAF*, an abstract argumentation framework
**Require:** *Prefs*, a set of preferences
**Ensure:** *AAF′*, an updated abstract argumentation framework

```
 1: function ApplyPreferences₂(AAF, Prefs)
 2:     for all (B, A) ∈ 𝓡 do
 3:         if A > B ∈ Prefs then                            ▷ A is preferred to its attacker B
 4:             𝓡′ ← 𝓡′ ∪ {(A, B)}                                          ▷ add reverse attack
 5:         else if A > B ∉ Prefs then                   ▷ A is not preferred to its attacker B
 6:             𝓡′ ← 𝓡′ ∪ {(B, A)}                                                 ▷ add attack
 7:         end if
 8:     end for
 9:     AAF′ ← (𝓐, 𝓡′)              ▷ argumentation framework AAF′ with attack relation 𝓡′
10:     return AAF′
11: end function
```

---

We establish that Algorithm 10 is sound (that is, its output is correct) and complete (that is, it outputs all solutions). We start with its soundness:

**Theorem 5.4.** *(Soundness): Algorithm 10 is sound in that given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets of preferences PrefSet, and semantics $\sigma$ as input, each preference set Prefs $\in$ PrefSet, when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics $\sigma$).*

*Proof.* We prove this by first proving the auxiliary Algorithm 11 for applying preferences. Algorithm 11 goes through all attacks $(B, A) \in \mathcal{R}$ in the input AAF and adds:

- the reverse attack $(A, B)$ to the the attack relation $\mathcal{R}'$ where the attacked argument $A$ is preferred to the attacking argument $B$ as per lines $3 - 4$.

- the attack $(B, A)$ to the the attack relation $\mathcal{R}'$ where the attacked argument $A$ is not preferred to the attacking argument $B$ as per lines $5 - 6$.

This ensures that any attacks of the form $(B, A)$ where the attacked argument $A$ is preferred to the attacking argument $B$ are not added to the attack relation $\mathcal{R}'$ and the reverse attacks of the form $(A, B)$ are added to the attack relation $\mathcal{R}'$. This results in the transformed $AAF' = (\mathcal{A}, \mathcal{R}')$.

Verification is then performed by invoking *ComputeExtensions* that computes and returns all the extensions in the transformed abstract argumentation framework $AAF'$ for a given semantics $\sigma$. Since, if the output is a single extension which is equal to the input extension as per lines $6 - 8$, therefore the *Prefs* is verified to be correct. Thus, *PrefSet* is then verified to be correct if the verification for each preference set *Prefs* $\in$ *PrefSet* is correct as per lines $10 - 14$.

$\square$

**Theorem 5.5.** *(Completeness): Algorithm 10 is complete in that given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets of preferences PrefSet, and semantics $\sigma$ as input, each Prefs $\in$ PrefSet is verified to be correct, i.e., each Prefs $\in$ PrefSet when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics $\sigma$).*

*Proof.* Algorithm 10 goes through each *Prefs* $\in$ *PrefSet* to verify its correctness, as per lines $3 - 9$. It then invokes Algorithm 11 to apply each *Prefs* to the *AAF* and get the updated *AAF'* as per line 4. *ComputeExtensions* is then invoked to get the extension of the updated *AAF'* under a given semantics $\sigma$.

Algorithm 11 goes through all attacks $(B, A) \in \mathcal{R}$ in the input AAF as per lines $2 - 8$ and adds:

- the reverse attack $(A, B)$ to the the attack relation $\mathcal{R}'$ where the attacked argument $A$ is preferred to the attacking argument $B$ as per lines $3 - 4$.

- the attack $(B, A)$ to the the attack relation $\mathcal{R}'$ where the attacked argument $A$ is not preferred to the attacking argument $B$ as per lines $5 - 6$.

Lines $10 - 14$ in Algorithm 8 ensure that all *Prefs* $\in$ *PrefSet* are verified to be correct. $\square$

After having proved the soundness and completeness of Algorithm 10, we establish its termination.

**Theorem 5.6.** *(Termination): Given an abstract argumentation framework AAF, an extension $\mathcal{E}$, a set of sets preferences PrefSet, and semantics $\sigma$ as input, Algorithm 10 always terminates.*

*Proof.* Algorithm 10 consists of one *for loop* that goes through each *Prefs* $\in$ *PrefSet* to verify its correctness, as per lines $3 - 9$. Since we assume that the input *PrefSet* is finite, therefore the *for loop* which iterates over all the elements *Prefs* $\in$ *PrefSet* in lines $3 - 9$ will always terminate.

Algorithm 10 invokes Algorithm 11 to apply each *Prefs* $\in$ *PrefSet* to the *AAF* and get the updated *AAF'*. We now prove that Algorithm 11 terminates. Since we assume that the attack relation $\mathcal{R}$ of the input *AAF* is finite, therefore the *for loop* which iterates over all the elements $(B, A) \in \mathcal{R}$ in lines $2 - 8$ will always terminate.

Thus, we have proved that Algorithm 10 always terminates. $\square$

## 6. Implementation and Evaluation

We now discuss the implementation details and evaluation of the algorithms presented in the previous sections.

### 6.1. Implementation

We have implemented our proposed algorithms[8] for evaluation purposes, in Java and using the Tweety library [40]. Figure 5 presents an overview of the system functionalities and information flow of the original approach.

The steps are as follows:

1. The **input** to the system is an abstract argumentation graph *AAF* (set of arguments $\mathcal{A}$ and attack relation $\mathcal{R}$ between arguments) and an extension $\mathcal{E}$ consisting of acceptable arguments. This is denoted by $(AAF, \mathcal{E})$.

---

[8]The source code of the implementation of all algorithms is available at https://github.com/Quratul-ain/AAF_Preferences
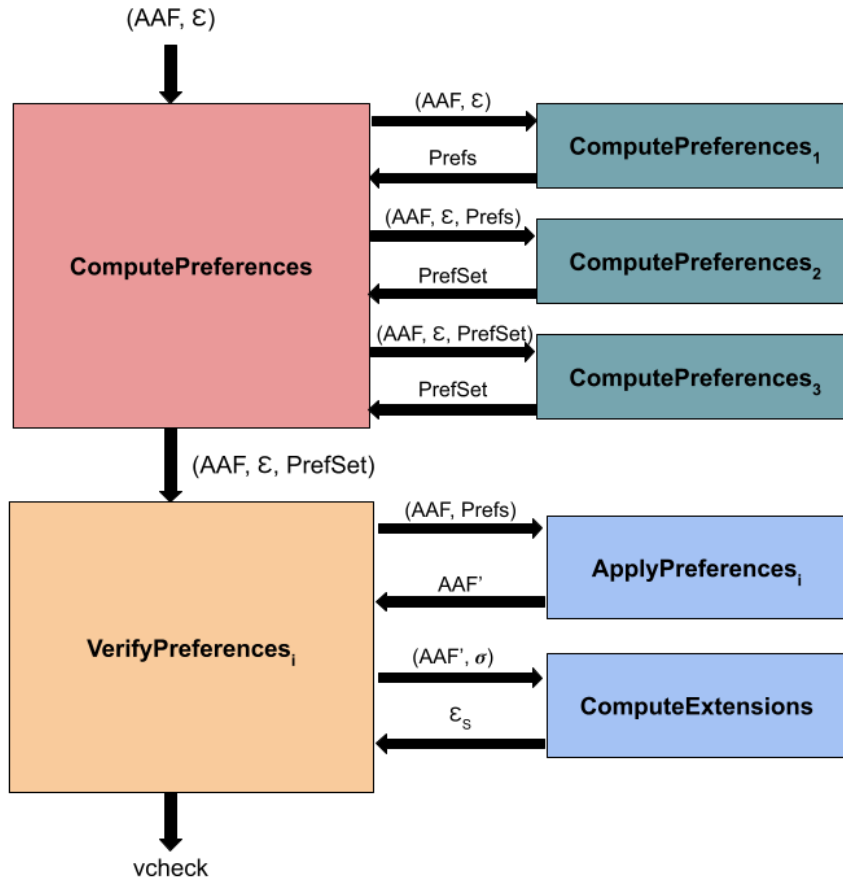
Figure 5: An overview of the system functionalities and information flow (Original Approach)

2. The **ComputePreferences** function:

    (i) invokes the **ComputePreferences$_1$** function (with input $(AAF, \mathcal{E})$) as given in Algorithm 2 that computes the set of case 1 preferences *Prefs*.

    (ii) invokes the **ComputePreferences$_2$** function (with input $(AAF, \mathcal{E}, Prefs)$) as given in Algorithm 3 that computes case 2 preferences and combines them with case 1 preferences. This results in *PrefSet* which is a set of sets of preferences.

    (iii) invokes the **ComputePreferences$_3$** function (with input $(AAF, \mathcal{E}, PrefSet)$) as given in Algorithm 4 that computes case 3 preferences and combines them with case 1 and case 2 preferences, i.e., *Prefs* and *PrefSet*. This results in an updated final *PrefSet* which is a set of sets of preferences containing all three cases of preferences combined together.

3. The **VerifyPreferences$_i$** function (with input $(AAF, \mathcal{E}, PrefSet)$):

    (i) invokes the **ApplyPreferences$_i$** function (with input $(AAF, Prefs)$, where $i = 1$ or $i = 2$ referring to the two methods of attack removal or reversal respectively) that applies the preferences to the abstract argumentation framework $AAF$ and returns an updated abstract argumentation framework $AAF'$.

    (ii) invokes the **ComputeExtensions** function that computes the set of extensions $\mathcal{E}_s$ and checks the conditions that $|\mathcal{E}_s = 1| \& \mathcal{E}' \in \mathcal{E}_s \ s.t. \ \mathcal{E}' = \mathcal{E}$. It returns a Boolean variable vcheck, which is *true* if the conditions are true for the preference sets; else it is *false*.
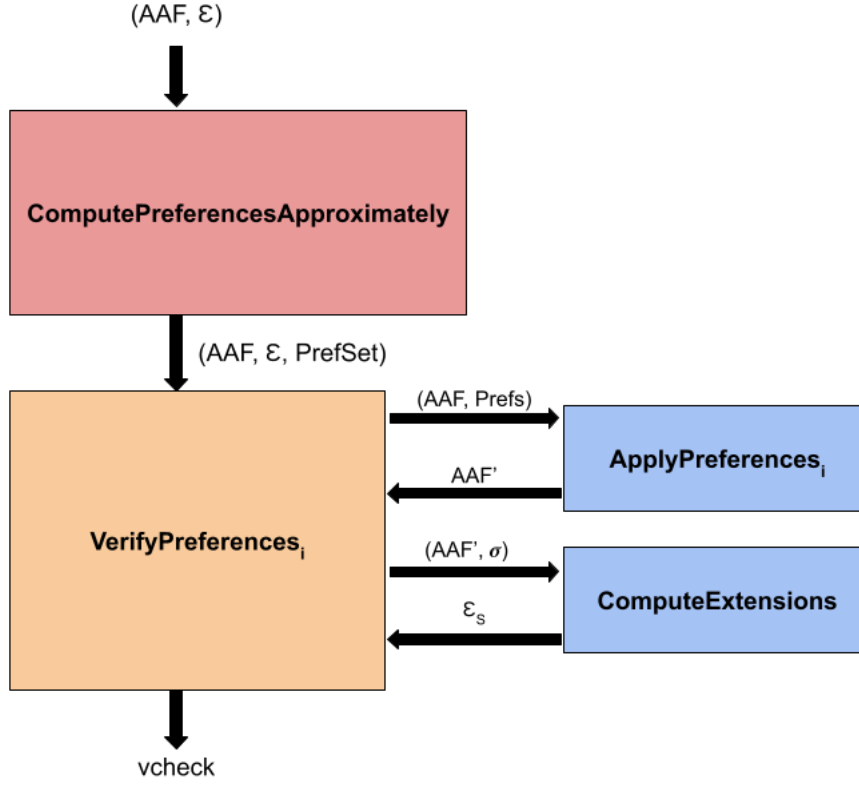
Figure 6: An overview of the system functionalities and information flow (Approximate Approach)

Figure 6 presents an overview of the system functionalities and information flow of the approximate approach. The steps are as follows:

1. The **input** to the system is an abstract argumentation graph *AAF* (set of arguments $\mathcal{A}$ and attack relation $\mathcal{R}$ between arguments) and an extension $\mathcal{E}$ consisting of acceptable arguments. This is denoted by $(AAF, \mathcal{E})$.

2. The **ComputePreferencesApproximately** function (with input $(AAF, \mathcal{E})$) as given in Algorithm 7 computes case 1, case 2 preferences randomly, and case 3 preferences randomly and combines them, which results in a set of preferences *Prefs* containing all three cases of preferences combined together.

3. *PrefSet* is created with a set of preferences *Prefs* from the previous step.

4. The **VerifyPreferences$_i$** function (with input $(AAF, \mathcal{E}, PrefSet)$):

   (i) invokes the **ApplyPreferences$_i$** function (with input $(AAF, Prefs)$, where $i = 1$ or $i = 2$ referring to the two methods of attack removal or reversal respectively) that applies the preferences to the abstract argumentation framework *AAF* and returns an updated abstract argumentation framework $AAF'$.

   (ii) invokes the **ComputeExtensions** function that computes the set of extensions $\mathcal{E}_s$ and checks the conditions that $|\mathcal{E}_s = 1|$ & $\mathcal{E}' \in \mathcal{E}_s$ *s.t.* $\mathcal{E}' = \mathcal{E}$. It returns a Boolean variable vcheck, which is *true* if the conditions are true for the preference sets; else it is *false*.

*6.2. Evaluation*

To evaluate how our algorithms perform, we carried out several experiments[9] to analyse various metrics of performance. In this section, we present the methodology for our experimental set-up, experiments performed and the

---

[9]All data sets are included in the appendix.

analysis of their results.

### 6.2.1. Methodology

We now present the experimental methodology that we have adopted in order to evaluate the algorithms used in our system as follows:

1. We generated abstract argumentation frameworks of increasing size, for which, we used an existing benchmark abstract argumentation framework generator from the Tweety library [40].

2. We input the number of arguments $AAF_{size}$ of the abstract argumentation framework and attack probability $Pr$ to the generator in order to randomly generate the abstract argumentation frameworks.

3. The attack probability $Pr$ is taken as $0.25, 0.50$ and $0.75$.

4. We then compute the grounded, preferred and stable extensions respectively. For each type of extension, a different set of experiments is performed.

5. For the grounded extension the value of $AAF_{size}$ starts at 4 and ends at $12$[10]. We only select the generated $AAF$ where the grounded extension is non-empty and at least of size 1. Additionally, we evaluated the approximate algorithm (with attack probability $Pr = 0.25$ in the AAF[11]) for the grounded extension with larger value of $AAF_{size}$ that starts at 5 and ends at 60.

6. For the preferred and stable extensions the value of $AAF_{size}$ starts at 4 and ends at $16$[12]. We select the extension of the largest size if there are more than two preferred extensions for the $AAF$ and similarly if there are more than two stable extensions for the $AAF$. Additionally, we evaluated the approximate algorithm for the preferred and stable extensions with larger value of $AAF_{size}$ that starts at 5 and ends at 60.

### 6.2.2. Experiments

The purpose of our experimental analysis is not only to present the scalability of the algorithms but also to analyse the effect of the attack probability on the computation and verification of preferences on the grounded, preferred and stable extensions. Furthermore, we experimentally check that our implementation reflects the soundness property (i.e., Theorem 5.4) proven before for the grounded, preferred and stable extensions, i.e., the correctness of all the computed preference sets. Full detail of the data sets generated for the experiments is given in Tables 4-28 in the Appendix. The experiments were run on an Apple Mac book Pro machine, with 16GB of memory and a 3.22 GHz, Apple M1 Pro 8-Core.

We investigate the following hypotheses for the original algorithm (i.e., Algorithm 1).

- *Hypothesis* 1. The computation of set of sets of preferences is low/high for the grounded, preferred and stable extensions with lower/higher probability of attacks in the AAF.

- *Hypothesis* 2. The number of sets of preferences grow exponentially with the increasing size of AAFs.

We investigate the following hypothesis for the approximate algorithm (i.e., Algorithm 7).

- *Hypothesis* 3. Algorithm 7 is scalable for increasing size of AAFs.

We investigate the following hypotheses for both original algorithm (i.e., Algorithm 1) and approximate algorithm (i.e., Algorithm 7).

- *Hypothesis* 4. The number of preferences in each preference set does not grow exponentially with the increasing size of AAFs.

---

[10]Please note, the original algorithm (i.e., Algorithm 1) becomes impractical after the maximum $AAF_{size}$ value 12.

[11]We were not able to generate AAFs of larger sizes with $Pr = 0.50$ and $Pr = 0.75$ that have at least one non-empty grounded extension using the existing benchmark abstract argumentation framework generator from the Tweety library [40]

[12]Please note, the original algorithm (i.e., Algorithm 1) becomes impractical after the maximum $AAF_{size}$ value 16.

- *Hypothesis* 5. The verification of preferences by *attack removal* approach has lower/higher computation run time for the grounded, preferred and stable extensions compared to the verification of preferences by *attack reversal* approach.

- *Hypothesis* 6. Algorithm 1 and Algorithm 7 hold the soundness property (i.e., Theorem 4) for the grounded, preferred and stable semantics.

For the *first* set of experiments we measured **the average execution time in milliseconds** (for 10 instances) between inputting an abstract argumentation framework (where attack probability $Pr$ is taken as $0.25, 0.50$ and $0.75$) and extension under the grounded, preferred and stable semantics, and computing all set of sets of preferences by the Original Algorithm (i.e. Algorithm 1) as shown in Figure 7a, Figure 11a and Figure 15a respectively, and by the Approximate Algorithm (i.e. Algorithm 7) as shown in Figure 8a, Figure 12a and Figure 16a respectively. We performed the following analysis:

1. For the **grounded extension**,

   - Original Algorithm: In Figure 7a, the line graph of the AAF with $Pr = 0.50$ shows a high computation time (worst-performance) at size 11 and falls down at size 12, where as with $Pr = 0.25$ computation time is a steady low (best-performance) and starts to increase at size 12, and for $Pr = 0.75$ shows an increase in computation at size 10 which starts to fall down at size 12.

   - Approximate Algorithm: In Figure 8a, the line graphs of the AAF with $Pr = 0.25, Pr = 0.50$, and $Pr = 0.75$ show a very low computation time that is not more than 1.0 ms for all AAF sizes.

2. For the **preferred extension**,

   - Original Algorithm: In Figure 11a, the line graph of the AAF with $Pr = 0.25$ shows a high computation time (worst-performance) which starts to increase at size 15, where as with $Pr = 0.75$ shows a steady low computation time (best-performance) till the largest size 16, and for $Pr = 0.50$ shows an increase in computation time at size 16.

   - Approximate Algorithm: In Figure 12a, the line graphs of the AAF with $Pr = 0.25, Pr = 0.50$, and $Pr = 0.75$ show a very low computation time that is not more than 1.0 ms for all AAF sizes.

3. For the **stable extension**,

   - Original Algorithm: In Figure 15a, the line graph of the AAF with $Pr = 0.25$ shows a high computation time (worst-performance) which starts to increase at size 15 and decreases slightly at size 16, where as with $Pr = 0.50$ and $Pr = 0.75$ shows a steady low computation time (best-performance) till the largest size 16.

   - Approximate Algorithm: In Figure 16a, the line graphs of the AAF with $Pr = 0.25, Pr = 0.50$, and $Pr = 0.75$ show a very low computation time that is not more than 1.0 ms for all AAF sizes.

*Hypothesis* 1. For the original algorithm (i.e., Algorithm 1), we conclude that when the number of attacks is less (i.e., $Pr = 0.25$) in the increasing sizes of AAF, the run time for computation of preferences for the grounded extension is lower compared to AAFs with higher number of attacks. On the other hand, we conclude that when the number of attacks is higher (i.e., $Pr = 0.75$) in the increasing sizes of AAF, the run time for computation of preferences for the preferred and stable extensions is lower compared to AAFs with lower number of attacks.

For the *second* set of experiments we measured **the average number of all the set of sets of preferences** (for 10 instances) computed for an input extension under the grounded, preferred and stable semantics for an abstract argumentation framework (where attack probability $Pr$ is taken as $0.25, 0.50$ and $0.75$) by the Original Algorithm (i.e. Algorithm 1) as shown in Figure 7b, Figure 11b and Figure 15b respectively. We performed the following analysis:

1. For the **grounded extension**, the line graph of the AAF (as shown in Figure 7b) with $Pr = 0.75$ shows a high number of preference sets at size 11 and falls down at size 12, where as with $Pr = 0.25$ the number of preference sets is a steady low and starts to increase at size 12, and for $Pr = 0.50$ the number of preference sets increases at size 9 and falls down at size 12.

2. For the **preferred extension**, the line graph of the AAF (as shown in Figure 11b) with $Pr = 0.50$ shows that the number of preference sets starts to increase at size 14, the line graph of the AAF with $Pr = 0.25$ shows that the number of preference sets starts to gradually increase at size 8 and grows rapidly at size 15, and the the line graph of the AAF with $Pr = 0.75$ shows a steady low number of preferences sets till the largest size 16.

3. For the **stable extension**, the line graph of the AAF (as shown in Figure 15b) with $Pr = 0.25$ shows that the number of preference sets starts to increase rapidly at size 15 and falls down slightly at size 16, the line graph of the AAF with $Pr = 0.50$ shows that the number of preference sets starts to increase at size 15, and the the line graph of the AAF with $Pr = 0.75$ shows a steady low number of preferences sets till the largest size 16.

*Hypothesis* 2. For the original algorithm (i.e., Algorithm 1), we conclude that the number of preference sets computed for the grounded extension is less when the AAFs of increasing sizes have lower number of attacks (i.e., $Pr = 0.25$) compared to AAFs with higher number of attacks. On the other hand, we conclude that the number of preference sets computed for the preferred and stable extensions is less when the AAFs of increasing sizes have higher number of attacks (i.e., $Pr = 0.75$) compared to AAFs with lower number of attacks. For all extensions, the growth of the number of sets of preferences is exponential with the increasing size of AAFs.

For the *third* set of experiments we measured **the average number of preferences in each set of preferences** (for 10 instances) computed for an input extension under the grounded, preferred and stable semantics for an abstract argumentation framework (where attack probability $Pr$ is taken as $0.25, 0.50$ and $0.75$) by the Original Algorithm (i.e. Algorithm 1) as shown in Figure 7c, Figure 11c and Figure 15c respectively, and by the Approximate Algorithm (i.e. Algorithm 7) as shown in Figure 8b, Figure 12b and Figure 16b respectively. We performed the following analysis:

1. For the **grounded extension**, as shown in Figure 7c and Figure 8b, the average number of preferences in each preference set does not grow more than 12 till AAF size 12. There is a steady increase in the number of preferences in each preference set with the increasing size of the AAF.

2. For the **preferred extension**, as shown in Figure 11c and Figure 12b, the average number of preferences in each preference set remains under 30 till AAF size 16. Similar to the grounded extension, there is a steady increase in the number of preferences in each preference set with the increasing size of the AAF.

3. For the **stable extension**, as shown in Figure 15c and Figure 16b, the average number of preferences in each preference set remains under 30 till AAF size 16. Similar to the grounded and preferred extensions, there is a steady increase in the number of preferences in each preference set with the increasing size of the AAF.

*Hypothesis* 4. For both the original algorithm (i.e., Algorithm 1) and approximate algorithm (i.e., Algorithm 7), we conclude that the number of preferences for all extensions increases steadily with the increasing size of the AAFs which is mostly similar for all attack probabilities. The growth in the number of preferences in each preference set for increasing size of the AAFs is not exponential.

Finally, for the *fourth* set of experiments we measured **the average elapsed time in milliseconds** (for 10 instances) between inputting an abstract argumentation framework (where attack probability $Pr$ is taken as $0.25, 0.50$ and $0.75$), extension under the grounded, preferred and stable semantics and the set of sets of preferences computed by the Original Algorithm (i.e. Algorithm 1) or the Approximate Algorithm (i.e. Algorithm 7), and verifying all the set of sets of preferences to be correct using the two verification methods[13]:

(i) *Attack removal*, as shown in Figure 9a, Figure 13a and Figure 17a for the Original Algorithm, and Figure 10a, Figure 14a and Figure 18a for the Approximate Algorithm.

(ii) *Attack reversal*, as shown in Figure 9b, Figure 13b and Figure 17b for the Original Algorithm, and Figure 10b, Figure 14b and Figure 18b for the Approximate Algorithm.

We performed the following analysis:

---

[13]This is done separately for both the Original and Approximate algorithms.

1. For the **grounded extension**,

   - For the preferences computed by the Original Algorithm: In Figure 9a and Figure 9b, the line graphs of the AAF with $Pr = 0.75$ for both verification approaches show a high computation time (worst-performance) at size 11 and this falls down at size 12, where as with $Pr = 0.25$ computation time is a steady low (best-performance) and starts to increase at size 12, and for $Pr = 0.50$ computation time increases at size 9 and falls down at size 12. Overall, although the line graph pattern of both verification methods is similar, the computation time of verification by attack removal is lower compared to the attack reversal verification.

   - For the preferences computed by the Approximate Algorithm: In Figure 10a and Figure 10b, the line graphs of the AAF with $Pr = 0.25, Pr = 0.50,$ and $Pr = 0.75$ for both verification approaches show a very low computation time that is not more than 1.0 ms for all AAF sizes.

2. For the **preferred extension**,

   - For the preferences computed by the Original Algorithm: In Figure 13a and Figure 13b, the line graphs of the AAF with $Pr = 0.25$ for both verification approaches show a high computation time (worst-performance) which starts to increase rapidly at size 15, where as with $Pr = 0.75$ shows a steady low computation time (best-performance) till the largest size 16, and with $Pr = 0.50$ the computation time goes up at size 14 and again at size 16 for both verification approaches. Overall, although the line graph pattern of both verification methods is similar, the computation time of verification by attack removal is lower compared to the attack reversal verification.

   - For the preferences computed by the Approximate Algorithm: In Figure 14a and Figure 14b, the line graph of the AAF with $Pr = 0.25, Pr = 0.50,$ and $Pr = 0.75$ for attack removal verification method shows a very low computation time that is not more than 5.0 ms for all AAF sizes, and for attack reversal verification method shows a very low computation time that is not more than 3.0 ms for all AAF sizes.

3. For the **stable extension**,

   - For the preferences computed by the Original Algorithm: In Figure 17a and Figure 17b, the line graphs of the AAF with $Pr = 0.25$ for both verification approaches show a high computation time (worst-performance) which starts to increase at size 15 and for the verification method of attack reversal goes down slightly at size 16, where as with $Pr = 0.75$ for both verification approaches show a steady low computation time (best-performance) till the largest size 16, and with $Pr = 0.50$ for both verification approaches show a small increase at size 15. Overall, although the line graph pattern of both verification methods is almost similar, the computation time of verification by attack reversal is lower compared to the attack removal verification.

   - For the preferences computed by the Approximate Algorithm: In Figure 18a and Figure 18b, the line graphs of the AAF with $Pr = 0.25, Pr = 0.50,$ and $Pr = 0.75$ for both verification approaches show a very low computation time that is not more than 4.0 ms for all AAF sizes.
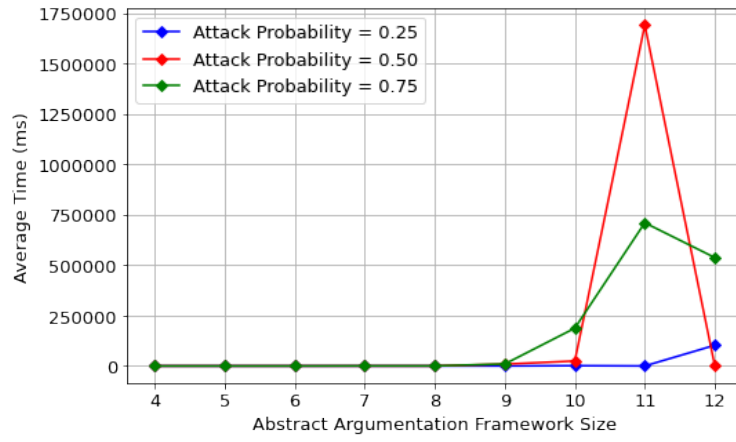
*Hypothesis* 5. Thus, for the original algorithm, we conclude that the verification of preferences by *attack removal* approach has lower run time for grounded and preferred extensions, whereas, the verification of preferences by *attack reversal* approach has lower run time for the stable extension. The line graph patterns are similar for both verification approaches, since this is dependent on the number of sets of preferences computed previously. For the approximate algorithm, we conclude that the run time for verification of preferences using both approaches is very low (only a few milliseconds).

*Hypothesis* 6. For both original algorithm (Algorithm 1) and approximate algorithm (Algorithm 7), both verification methods resulted in the original input extension for all semantics (i.e., grounded, preferred and stable). Thus, we conclude that Algorithm 1 and Algorithm 7 hold the soundness property for the grounded, preferred and stable semantics in our experimental evaluation.
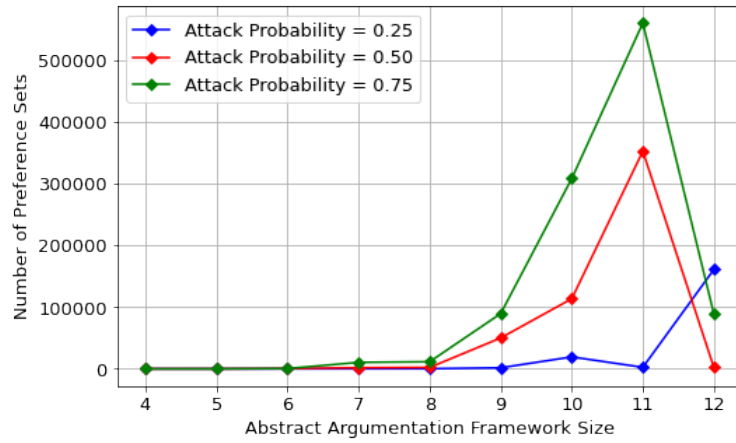
From the above experimental analysis it is clear that the original Algorithm 1 is exponential in complexity due to the exponential growth of sets of preferences for larger AAF sizes, however, it is useful to identify the maximum AAF size at which point this becomes impractical, which is AAF size 12 for the grounded extension and AAF size 16

for the preferred and stable extensions as shown in the experiments. The approximate Algorithm 7 has a very low run time (i.e., just a few milliseconds) and will be suited for real world settings where scalability of AAF size is important.

*Hypothesis* 3. We carried out further experiments to evaluate the scalability of the approximate Algorithm 7 for computing preferences and Algorithms 8 and 10 for verifying preferences on larger AAF sizes (i.e., sizes between 5 to 60) as shown in Figures 19, 20, 21, 22, 23 and 24. It is evident from the figures that the approximate Algorithm 7, and Algorithms 8 and 10 for verifying preferences have very low run time (i.e., just a few milliseconds) even for larger AAF sizes.
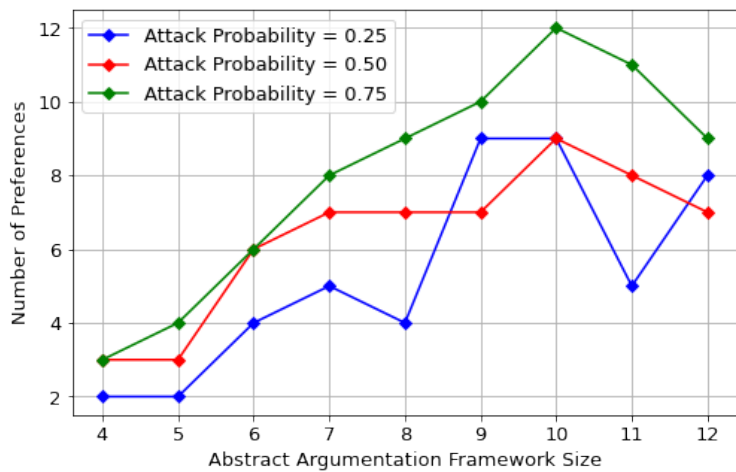
Figure 7: Computing Preferences of the Grounded Extension (Original Algorithm)



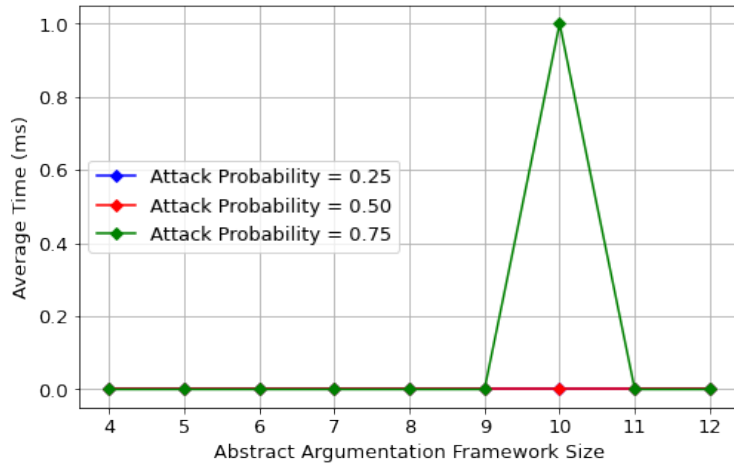(a) Average time in milliseconds for computing all preference sets
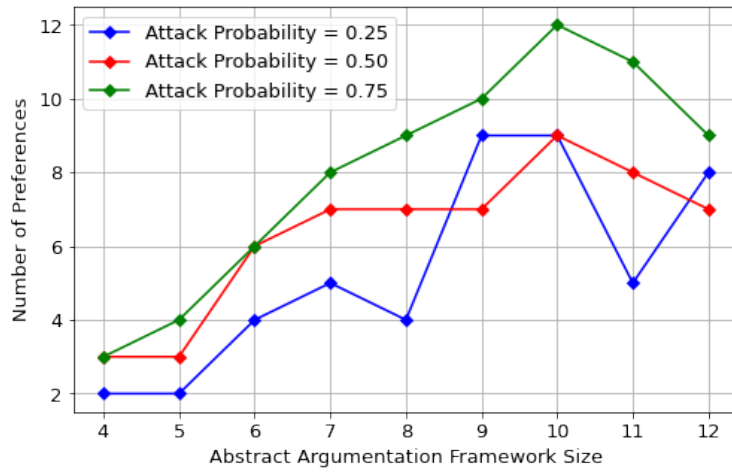


(b) Average number of preference sets



(c) Average number of preferences in each preference set

Figure 8: Computing Preferences of the Grounded Extension (Approximate Algorithm)
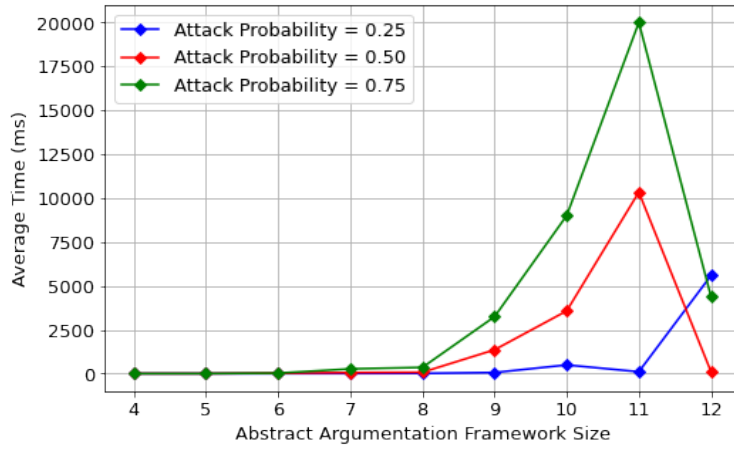


(a) Average time in milliseconds for computing all preference sets
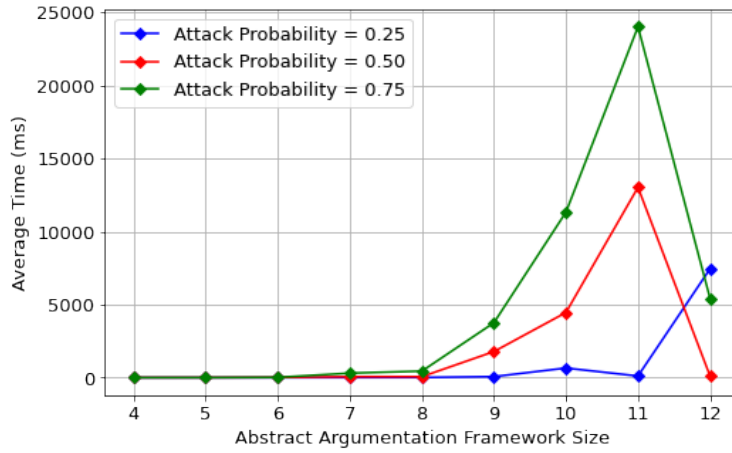


(b) Average number of preferences in each preference set

Figure 9: Verifying Preferences of the Grounded Extension (Original Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

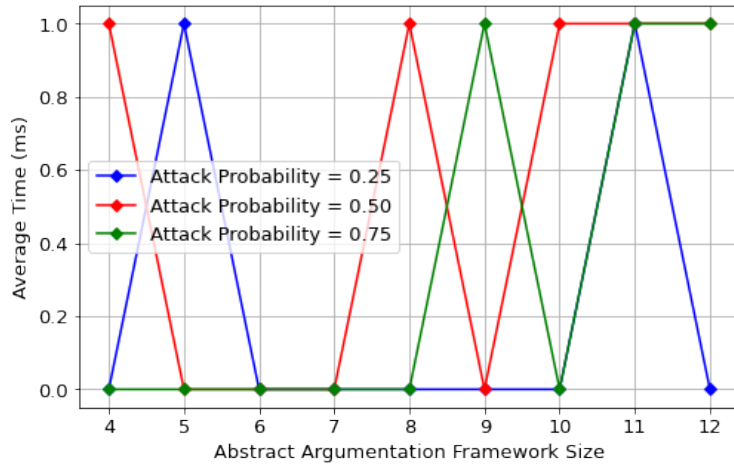Figure 10: Verifying Preferences of the Grounded Extension (Approximate Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 11: Computing Preferences of the Preferred Extension (Original Algorithm)



(a) Average time in milliseconds for computing all preference sets



(b) Average number of preference sets



(c) Average number of preferences in each preference set

Figure 12: Computing Preferences of the Preferred Extension (Approximate Algorithm)



(a) Average time in milliseconds for computing all preference sets



(b) Average number of preferences in each preference set

Figure 13: Verifying Preferences of the Preferred Extension (Original Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)
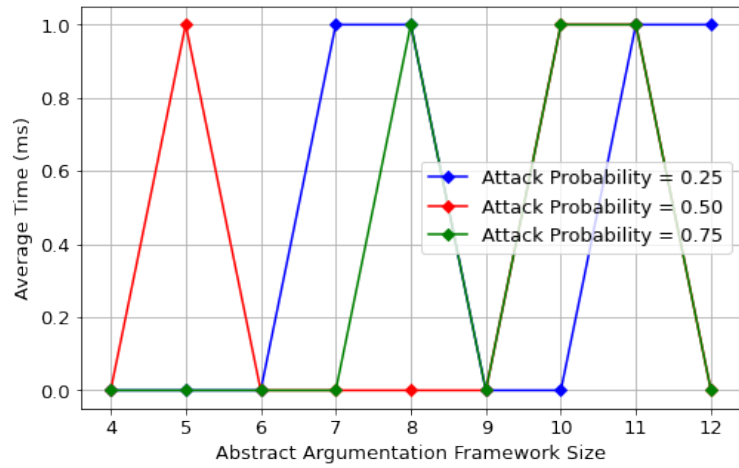


(b) Average time in milliseconds for verifying all preference sets (attack reversal)

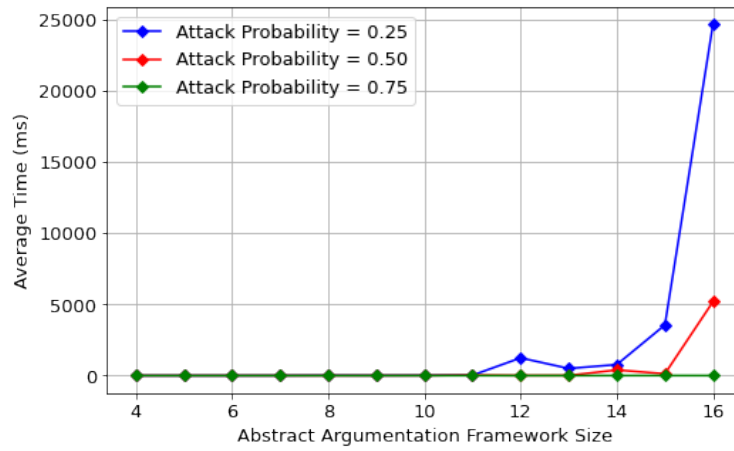Figure 14: Verifying Preferences of the Preferred Extension (Approximate Algorithm)



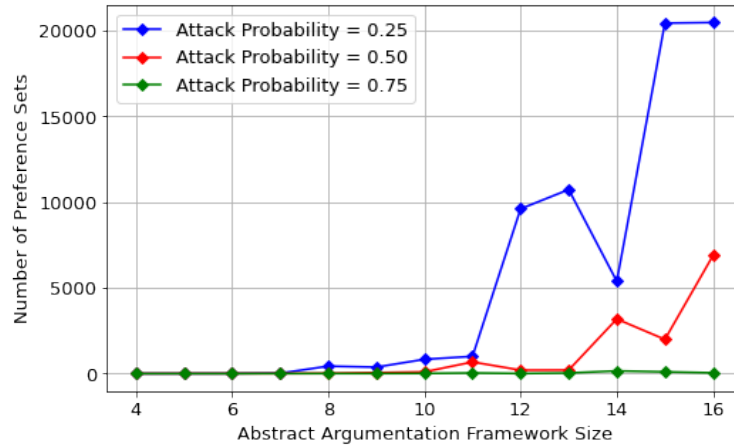(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 15: Computing Preferences of the Stable Extension (Original Algorithm)



(a) Average time in milliseconds for computing all preference sets



(b) Average number of preference sets



(c) Average number of preferences in each preference set

Figure 16: Computing Preferences of the Stable Extension (Approximate Algorithm)



(a) Average time in milliseconds for computing all preference sets



(b) Average number of preferences in each preference set

Figure 17: Verifying Preferences of the Stable Extension (Original Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)



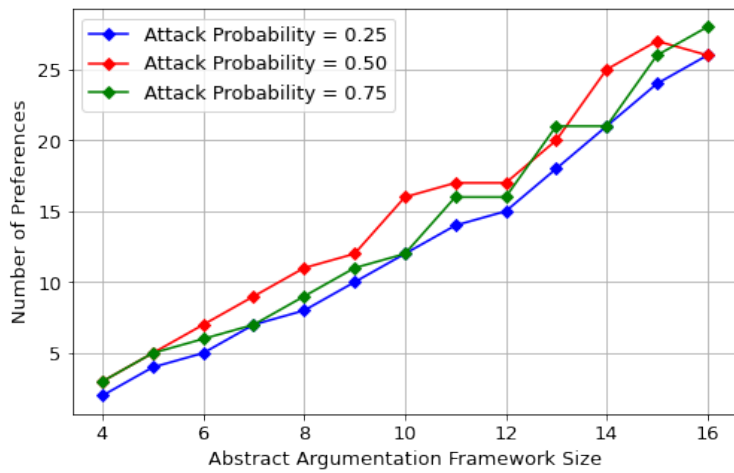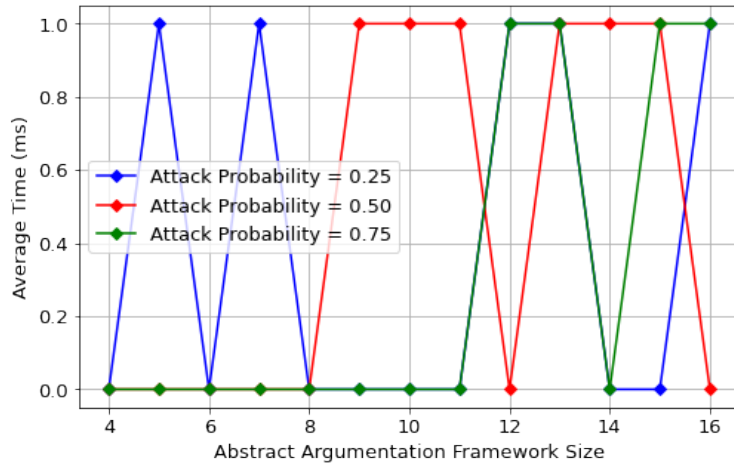(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 18: Verifying Preferences of the Stable Extension (Approximate Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 19: Computing Preferences of the Grounded Extension for larger AAF Sizes (Approximate Algorithm)



(a) Average time in milliseconds for computing all preference sets
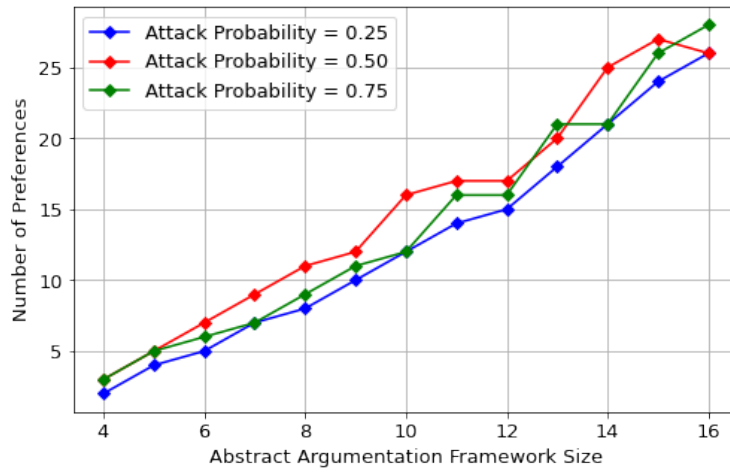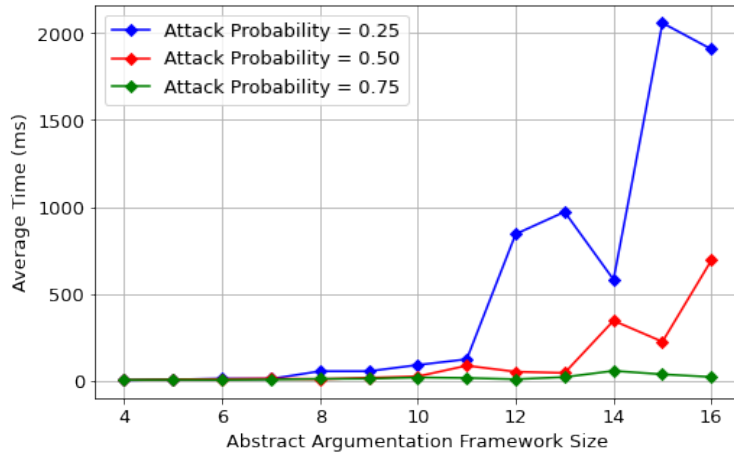


(b) Average number of preferences in each preference set

Figure 20: Verifying Preferences of the Grounded Extension for larger AAF Sizes (Approximate Algorithm)



(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 21: Computing Preferences of the Preferred Extension for larger AAF Sizes (Approximate Algorithm)
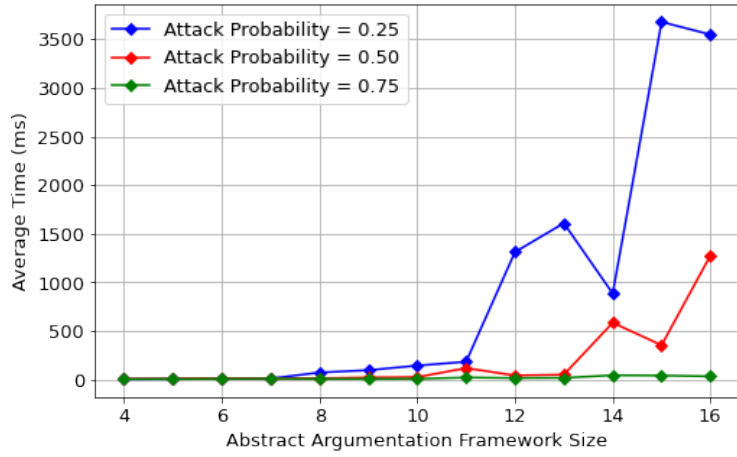


(a) Average time in milliseconds for computing all preference sets



(b) Average number of preferences in each preference set

Figure 22: Verifying Preferences of the Preferred Extension for larger AAF Sizes (Approximate Algorithm)
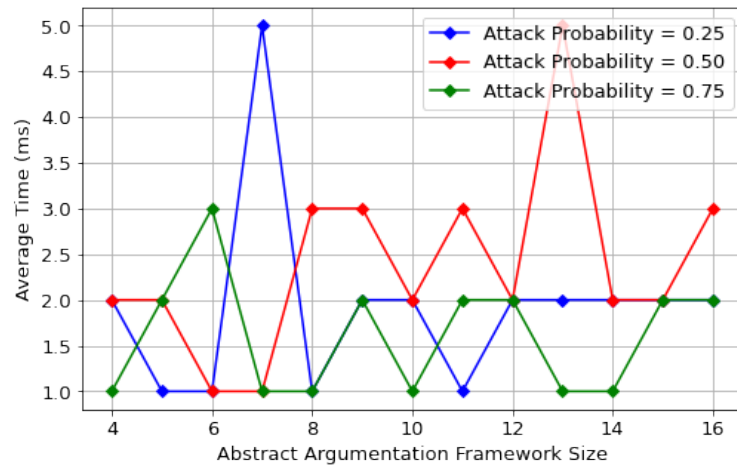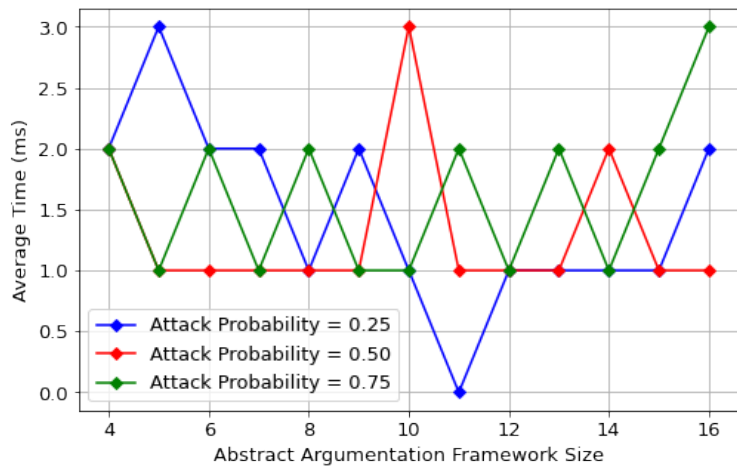


(a) Average time in milliseconds for verifying all preference sets (attack removal)



(b) Average time in milliseconds for verifying all preference sets (attack reversal)

Figure 23: Computing Preferences of the Stable Extension for larger AAF Sizes (Approximate Algorithm)



(a) Average time in milliseconds for computing all preference sets



(b) Average number of preferences in each preference set

Figure 24: Verifying Preferences of the Stable Extension for larger AAF Sizes (Approximate Algorithm)
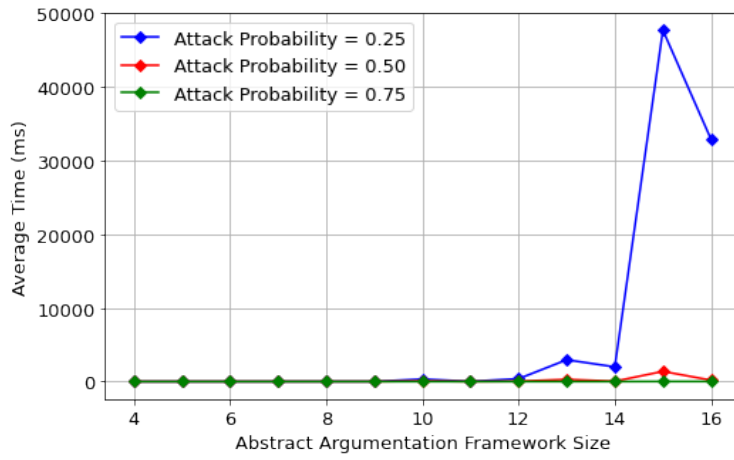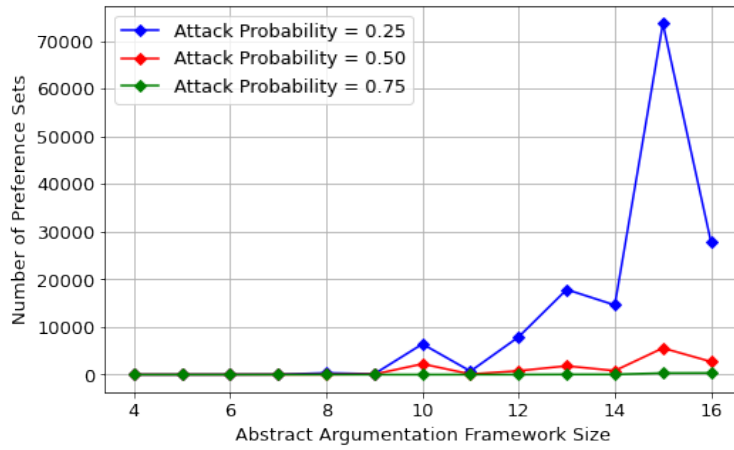


(a) Average time in milliseconds for verifying all preference sets (attack removal)



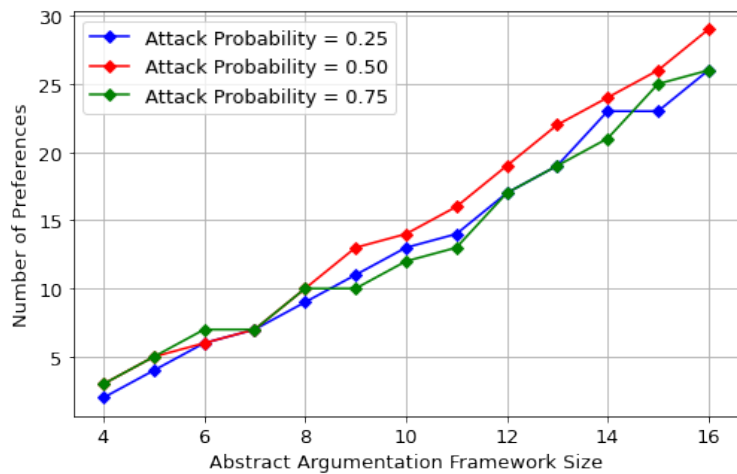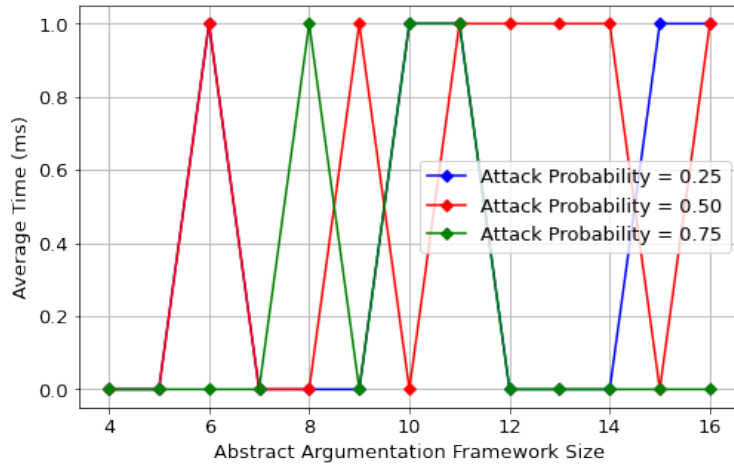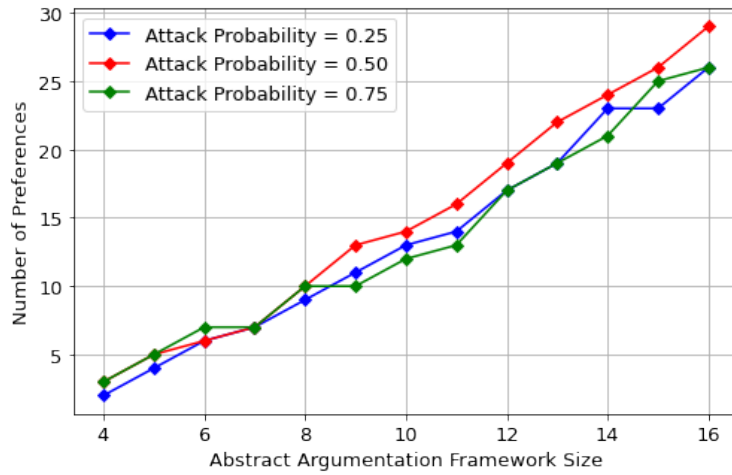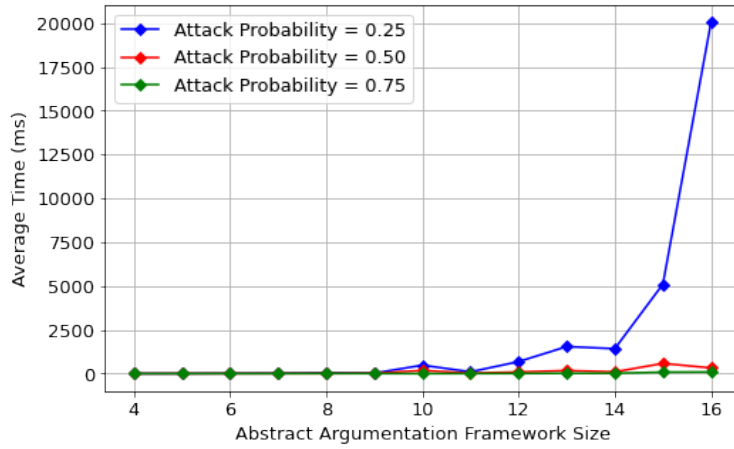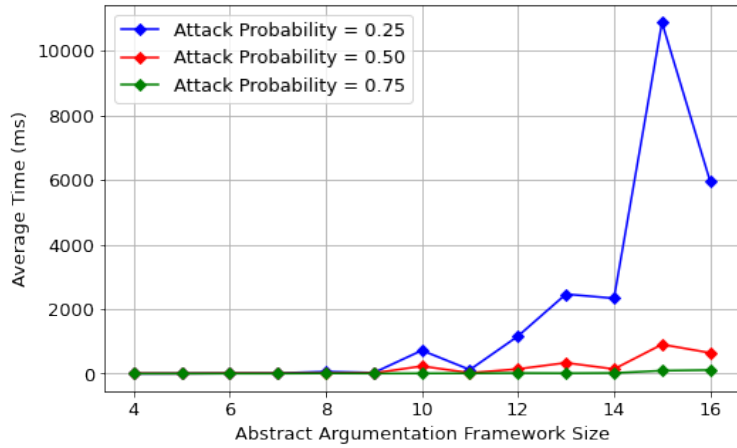(b) Average time in milliseconds for verifying all preference sets (attack reversal)

## 7. Conclusions and Future Work

In this paper we have described a novel extension-based approach to compute and verify (i.e., assess) abstract argument preferences. We have presented a novel algorithm that takes an abstract argumentation framework and a set of conflict-free arguments (extension) as input and computes all possible sets of preferences (restricted to three identified cases) that are valid for the acceptability of the arguments in the input extension. We have shown that the complexity of computing sets of preferences is exponential in the number of arguments, and thus, describe a novel approximate approach and algorithm to compute the preferences that is scalable. Furthermore, we have presented novel algorithms for verifying the computed sets of preferences, ensuring their validity. We have experimentally shown that both approaches, i.e., attack removal and reversal for the verification, output the desired input extension. We have implemented all the algorithms and have build a complete system for computing and verifying preferences. We have performed various experiments for the grounded, preferred and stable semantics with different attack probabilities in the abstract argumentation frameworks to evaluate the algorithms and performed an analysis of the results obtained.

This work has applications in decision support systems [5, 6, 41] and recommender systems [7], where the resulting decision(s) or recommendation(s) can be justified by the preference set(s). Another application would be to

explore our work in dialogue strategies [8, 9], for instance in computational persuasion [10, 11] or negotiation [12] – where an agent may have the capability of inferring preferences and reach her goal if (s)he enforces at least one of several desired sets of arguments with the application of preferences. The inferred preferences, in particular the unique and common preferences can be exploited in optimizing the choice of move in persuasion dialogues for behaviour change as well as in negotiation dialogues to reach agreement.

As future work, we plan to investigate different ways to aggregate and assess the sets of preferences. Additionally, we also plan to do an empirical evaluation of our proposed work on concrete examples. This will allow us to filter sets of preferences, i.e., to accept or reject them; or to rank the sets of preferences by human participants. Furthermore, we plan to extend our approach to value-based argumentation frameworks [27], where it would be interesting to elicit values that the arguments promote or support to determine preferences over arguments. An extension of our work [26] to assumption-based argumentation frameworks ABA [35, 36] and ABA$^+$ [30] has been presented in [34], where we were interested in eliciting preferences at the assumption level, however in the future, we aim to go beyond this by exploring other structured argumentation frameworks [42, 32], in particular ASPIC$^+$ [32]. Furthermore, we intend to investigate the relationship between extension enforcement [43, 44] and our work.

## Acknowledgements

## References

[1] G. Pigozzi, A. Tsoukiàs, P. Viappiani, Preferences in artificial intelligence, Annals of Mathematics and Artificial Intelligence 77 (3) (2016) 361–401.

[2] T. Walsh, Representing and reasoning with preferences, AI Magazine 28 (4) (2007) 59–70.

[3] K. Konczak, Voting procedures with incomplete preferences, in: in Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling, 2005.

[4] M. Pini, F. Rossi, K. Venable, T. Walsh, Incompleteness and incomparability in preference aggregation: Complexity results, Artificial Intelligence 175 (7) (2011) 1272 – 1289.

[5] R. H. Sprague, Jr., H. J. Watson (Eds.), Decision Support Systems (3rd Ed.): Putting Theory into Practice, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[6] Q. Mahesar, V. Dimitrova, D. R. Magee, A. G. Cohn, Uncertainty management for rule-based decision support systems, in: 29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017, IEEE Computer Society, 2017, pp. 884–891.

[7] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, Recommender Systems Handbook, 1st Edition, Springer-Verlag, 2010.

[8] M. Thimm, Strategic argumentation in multi-agent systems, KI 28 (3) (2014) 159–168.

[9] T. Rienstra, M. Thimm, N. Oren, Opponent models with uncertainty for strategic argumentation, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI/AAAI, 2013, pp. 332–338.

[10] A. Hunter, Towards a framework for computational persuasion with applications in behaviour change, Argument Comput. 9 (1) (2018) 15–40.

[11] E. Hadoux, A. Hunter, S. Polberg, Strategic argumentation dialogues for persuasion: Framework and experiments based on modelling the beliefs and concerns of the persuadee, Argument Comput. 14 (2) (2023) 109–161.

[12] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, Knowl. Eng. Rev. 18 (4) (2003) 343–375.

[13] P. Besnard, A. Hunter, A logic-based theory of deductive arguments, Artificial Intelligence 128 (1) (2001) 203 – 235.

[14] A. J. García, G. R. Simari, Defeasible logic programming: An argumentative approach, Theory Pract. Log. Program. 4 (2) (2004) 95–138.

[15] G. R. Simari, R. P. Loui, A mathematical treatment of defeasible reasoning and its implementation, Artificial Intelligence 53 (2) (1992) 125 – 157.

[16] L. Amgoud, H. Prade, Using arguments for making and explaining decisions, Artificial Intelligence 173 (3) (2009) 413 – 436.

[17] B. Bonet, H. Geffner, Arguing for decisions: A qualitative model of decision making, in: Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence, UAI'96, Morgan Kaufmann Publishers Inc., 1996, pp. 98–105.

[18] J. Muller, A. Hunter, An argumentation-based approach for decision making, in: Proceedings of the 2012 IEEE 24th International Conference on Tools with Artificial Intelligence - Volume 01, ICTAI '12, IEEE Computer Society, 2012, pp. 564–571.

[19] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.

[20] L. Amgoud, C. Cayrol, On the acceptability of arguments in preference-based argumentation, in: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI'98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 1–7.

[21] S. Modgil, Reasoning about preferences in argumentation frameworks, Artificial Intelligence 173 (9) (2009) 901 – 934.

[22] H. Prakken, G. Sartor, Argument-based extended logic programming with defeasible priorities, Journal of Applied Non-Classical Logics 7 (1997) 25–75.

[23] L. Amgoud, C. Cayrol, A reasoning model based on the production of acceptable arguments, Annals of Mathematics and Artificial Intelligence 34 (1) (2002) 197–215.

[24] L. Amgoud, S. Vesic, Rich preference-based argumentation frameworks, International Journal of Approximate Reasoning 55 (2) (2014) 585 – 606.

[25] S. Kaci, L. van der Torre, Preference-based argumentation: Arguments supporting multiple values, Int. J. Approx. Reasoning 48 (3) (2008) 730–751.

[26] Q. Mahesar, N. Oren, W. W. Vasconcelos, Computing preferences in abstract argumentation, in: PRIMA 2018: Principles and Practice of Multi-Agent Systems - 21st International Conference, Tokyo, Japan, October 29 - November 2, 2018, Proceedings, Vol. 11224 of Lecture Notes in Computer Science, Springer, 2018, pp. 387–402.

[27] T. J. M. Bench-Capon, Persuasion in practical argument using value-based argumentation frameworks, Journal of Logic and Computation 13 (3) (2003) 429–448.

[28] M. Caminada, L. Amgoud, On the evaluation of argumentation formalisms, Artificial Intelligence 171 (5) (2007) 286 – 310.

[29] L. Amgoud, S. Vesic, A new approach for preference-based argumentation frameworks, Annals of Mathematics and Artificial Intelligence 63 (2) (2011) 149–183.

[30] K. Cyras, F. Toni, ABA+: assumption-based argumentation with preferences, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR, 2016, pp. 553–556.

[31] T. Wakaki, Assumption-based argumentation equipped with preferences and its application to decision making, practical reasoning, and epistemic reasoning, Comput. Intell. 33 (4) (2017) 706–736.

[32] S. Modgil, H. Prakken, The $ASPIC^+$ framework for structured argumentation: a tutorial, Argument & Computation 5 (1) (2014) 31–62.

[33] P. Besnard, A. Hunter, Constructing argument graphs with deductive arguments: a tutorial, Argument & Computation 5 (1) (2014) 5–30.

[34] Q. Mahesar, N. Oren, W. W. Vasconcelos, Preference elicitation in assumption-based argumentation, in: PRIMA 2020: Principles and Practice of Multi-Agent Systems - 23rd International Conference, Nagoya, Japan, November 18-20, 2020, Proceedings, Vol. 12568 of Lecture Notes in Computer Science, Springer, 2020, pp. 199–214.

[35] P. M. Dung, R. A. Kowalski, F. Toni, Assumption-Based Argumentation, Springer US, 2009.

[36] F. Toni, A tutorial on assumption-based argumentation, Argument & Computation 5 (1) (2014) 89–117.

[37] N. D. Hung, V. Huynh, Revealed preference in argumentation: Algorithms and applications, Int. J. Approx. Reason. 131 (2021) 214–251.

[38] S. Benferhat, D. Dubois, H. Prade, Argumentative inference in uncertain and inconsistent knowledge bases, in: D. Heckerman, A. Mamdani (Eds.), Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1993, pp. 411 – 419.

[39] C. Cayrol, V. Royer, C. Saurel, Management of preferences in assumption-based reasoning, in: B. Bouchon-Meunier, L. Valverde, R. R. Yager (Eds.), IPMU '92—Advanced Methods in Artificial Intelligence, Springer Berlin Heidelberg, 1993, pp. 13–22.

[40] M. Thimm, The formal argumentation libraries of tweety, in: Proc. of the International Workshop on Theory and Applications of Formal Argument, 2017.

[41] L. Wei, H. Du, Q. Mahesar, K. A. Ammari, D. R. Magee, B. Clarke, V. Dimitrova, D. Gunn, D. Entwisle, H. Reeves, A. G. Cohn, A decision support system for urban infrastructure inter-asset management employing domain ontologies and qualitative uncertainty-based reasoning, Expert Syst. Appl. 158 (2020) 113461.

[42] P. Besnard, A. Garcia, A. Hunter, S. Modgil, H. Prakken, G. Simari, F. Toni, Introduction to structured argumentation, Argument and Computation 5 (1) (2014) 1–4.

[43] R. Baumann, What does it take to enforce an argument? minimal change in abstract argumentation, in: Proceedings of the 20th European Conference on Artificial Intelligence, 2012, pp. 127–132.

[44] S. Coste-Marquis, S. Konieczny, J.-G. Mailly, P. Marquis, Extension enforcement in abstract argumentation as an optimization problem, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015, pp. 2876–2882.

# Appendix A. Preference Sets

Table A.3: Preference sets for all conflict-free extensions

| Conflict-free Extensions | Preference Sets |
|---|---|
| $\{A, C, E\}$ | $\{\{C > D, A > B, C > B, E > D\}$ <br> $\{C > D, A > B, C > B, E = D\}$ <br> $\{C > D, A > B, C > B, D > E\}$ <br> $\{C > D, A > B, C = B, E > D\}$ <br> $\{C > D, A > B, C = B, E = D\}$ <br> $\{C > D, A > B, C = B, D > E\}$ <br> $\{C > D, A = B, C > B, E > D\}$ <br> $\{C > D, A = B, C > B, E = D\}$ <br> $\{C > D, A = B, C > B, D > E\}$ <br> $\{C > D, A = B, C = B, E > D\}$ <br> $\{C > D, A = B, C = B, E = D\}$ <br> $\{C > D, A = B, C = B, D > E\}\}$ |
| $\{A, D\}$ | $\{\{D > C, A > B, D > E\}$ <br> $\{D > C, A > B, D = E\}$ <br> $\{D > C, A = B, D > E\}$ <br> $\{D > C, A = B, D = E\}\}$ |
| $\{B, D\}$ | $\{\{B > A, B > C, D > C, D > E\}$ <br> $\{B > A, B > C, D > C, D = E\}\}$ |
| $\{A, C\}$ | $\{\{C > D, A > B, C > B\}$ <br> $\{C > D, A > B, C = B\}$ <br> $\{C > D, A = B, C > B\}$ <br> $\{C > D, A = B, C = B\}\}$ |
| $\{A, E\}$ | $\{\{E > D, A > B\}$ <br> $\{E > D, A = B\}\}$ |
| $\{B, E\}$ | $\{\{B > A, B > C, E > D\}\}$ |
| $\{C, E\}$ | $\{\{C > D, C > B, E > D\}$ <br> $\{C > D, C > B, E = D\}$ <br> $\{C > D, C > B, D > E\}$ <br> $\{C > D, C = B, E > D\}$ <br> $\{C > D, C = B, E = D\}$ <br> $\{C > D, C = B, D > E\}\}$ |
| $\{A\}$ | $\{\{A > B\}$ <br> $\{A = B\}\}$ |
| $\{B\}$ | $\{\{B > A, B > C\}\}$ |
| $\{C\}$ | $\{\{C > D, C > B\}$ <br> $\{C > D, C = B\}\}$ |
| $\{D\}$ | $\{\{D > C, D > E\}$ <br> $\{D > C, D = E\}\}$ |
| $\{E\}$ | $\{\{E > D\}\}$ |
| $\emptyset$ | $\emptyset$ |

## Appendix B. Data sets generated for the Experimental Analysis

We present below all the data sets in the form of tables that were generated to conduct the experimental analysis presented in Section 6.2. The abbreviations used in the titles of columns in the following tables are given as follows:

- **AAF Size:** Abstract Argumentation Framework Size.

- **Ext Size:** Extension Size.

- **Attacks:** Number of Attacks.

- **Preference Sets:** Number of Preference Sets.

- **Preferences:** Number of Preferences.

- **CTime:** Time for Computing all Preference Sets in milliseconds.

- **VTime1:** Time for Verifying all Preference Sets in milliseconds, using Attack Removal Method.

- **VTime2:** Time for Verifying all Preference Sets in milliseconds, using Attack Reversal Method.

## Data sets for the Original Algorithm

Table B.4: Grounded Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 8 | 2 | 0 | 3 | 4 |
| 5 | 3 | 3 | 12 | 2 | 1 | 4 | 5 |
| 6 | 3 | 6 | 51 | 4 | 3 | 7 | 8 |
| 7 | 2 | 10 | 159 | 5 | 4 | 11 | 14 |
| 8 | 2 | 12 | 242 | 4 | 6 | 17 | 18 |
| 9 | 4 | 16 | 1726 | 9 | 27 | 65 | 76 |
| 10 | 3 | 22 | 19354 | 9 | 1452 | 497 | 663 |
| 11 | 2 | 27 | 2500 | 5 | 130 | 111 | 118 |
| 12 | 3 | 30 | 160976 | 8 | 102678 | 5589 | 7450 |

Table B.5: Grounded Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 5 | 18 | 3 | 1 | 4 | 7 |
| 5 | 2 | 8 | 21 | 3 | 1 | 4 | 8 |
| 6 | 2 | 13 | 408 | 6 | 7 | 27 | 19 |
| 7 | 2 | 19 | 1869 | 7 | 40 | 61 | 69 |
| 8 | 2 | 25 | 2279 | 7 | 40 | 80 | 71 |
| 9 | 2 | 29 | 50088 | 7 | 9163 | 1361 | 1789 |
| 10 | 2 | 41 | 113597 | 9 | 24540 | 3574 | 4461 |
| 11 | 2 | 52 | 352546 | 8 | 1689643 | 10334 | 13022 |
| 12 | 1 | 60 | 2190 | 7 | 59 | 120 | 124 |

Table B.6: Grounded Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 19 | 3 | 1 | 5 | 7 |
| 5 | 1 | 13 | 52 | 4 | 2 | 7 | 9 |
| 6 | 2 | 19 | 509 | 6 | 9 | 32 | 24 |
| 7 | 2 | 28 | 10506 | 8 | 481 | 273 | 315 |
| 8 | 2 | 36 | 11539 | 9 | 336 | 358 | 452 |
| 9 | 2 | 49 | 89555 | 10 | 9906 | 3245 | 3760 |
| 10 | 2 | 61 | 308365 | 12 | 187235 | 9002 | 11324 |
| 11 | 1 | 75 | 559968 | 11 | 709698 | 19996 | 24010 |
| 12 | 1 | 87 | 88733 | 9 | 537216 | 4400 | 5438 |

Table B.7: Preferred Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 4 | 2 | 2 | 0 | 1 | 3 |
| 5 | 2 | 7 | 4 | 4 | 1 | 5 | 5 |
| 6 | 3 | 8 | 8 | 5 | 1 | 10 | 8 |
| 7 | 3 | 12 | 25 | 7 | 3 | 9 | 11 |
| 8 | 3 | 15 | 437 | 8 | 12 | 53 | 72 |
| 9 | 3 | 20 | 380 | 10 | 7 | 53 | 96 |
| 10 | 4 | 21 | 837 | 12 | 13 | 89 | 143 |
| 11 | 4 | 27 | 1008 | 14 | 28 | 122 | 183 |
| 12 | 4 | 32 | 9613 | 15 | 1246 | 843 | 1311 |
| 13 | 4 | 41 | 10736 | 18 | 501 | 971 | 1608 |
| 14 | 4 | 49 | 5389 | 21 | 774 | 581 | 885 |
| 15 | 4 | 52 | 20442 | 24 | 3554 | 2059 | 3677 |
| 16 | 5 | 60 | 20480 | 26 | 24654 | 1908 | 3548 |

Table B.8: Preferred Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 3 | 3 | 0 | 2 | 6 |
| 5 | 2 | 10 | 6 | 5 | 1 | 6 | 9 |
| 6 | 2 | 15 | 7 | 7 | 1 | 8 | 12 |
| 7 | 2 | 22 | 10 | 9 | 1 | 11 | 10 |
| 8 | 2 | 28 | 17 | 11 | 1 | 7 | 13 |
| 9 | 2 | 38 | 60 | 12 | 3 | 15 | 21 |
| 10 | 3 | 45 | 102 | 16 | 5 | 23 | 25 |
| 11 | 2 | 56 | 682 | 17 | 32 | 85 | 117 |
| 12 | 2 | 66 | 204 | 17 | 8 | 50 | 42 |
| 13 | 2 | 81 | 210 | 20 | 9 | 43 | 50 |
| 14 | 3 | 90 | 3189 | 25 | 399 | 344 | 584 |
| 15 | 3 | 105 | 1997 | 27 | 129 | 223 | 353 |
| 16 | 2 | 122 | 6886 | 26 | 5190 | 692 | 1276 |

Table B.9: Preferred Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 1 | 9 | 2 | 3 | 1 | 3 | 5 |
| 5 | 1 | 15 | 3 | 5 | 0 | 4 | 5 |
| 6 | 1 | 23 | 3 | 6 | 0 | 3 | 7 |
| 7 | 1 | 32 | 7 | 7 | 1 | 5 | 9 |
| 8 | 1 | 42 | 9 | 9 | 1 | 9 | 7 |
| 9 | 2 | 54 | 9 | 11 | 1 | 10 | 7 |
| 10 | 2 | 67 | 14 | 12 | 2 | 17 | 7 |
| 11 | 2 | 81 | 41 | 16 | 3 | 14 | 21 |
| 12 | 2 | 99 | 9 | 16 | 1 | 7 | 15 |
| 13 | 2 | 115 | 40 | 21 | 2 | 19 | 17 |
| 14 | 2 | 134 | 161 | 21 | 6 | 55 | 44 |
| 15 | 2 | 157 | 106 | 26 | 5 | 35 | 41 |
| 16 | 2 | 184 | 48 | 28 | 3 | 20 | 33 |

Table B.10: Stable Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 4 | 2 | 2 | 0 | 3 | 1 |
| 5 | 3 | 6 | 4 | 4 | 1 | 4 | 4 |
| 6 | 3 | 10 | 17 | 6 | 1 | 12 | 10 |
| 7 | 3 | 11 | 40 | 7 | 2 | 15 | 14 |
| 8 | 4 | 14 | 346 | 9 | 7 | 42 | 62 |
| 9 | 4 | 19 | 144 | 11 | 5 | 30 | 29 |
| 10 | 4 | 22 | 6426 | 13 | 330 | 483 | 720 |
| 11 | 4 | 29 | 715 | 14 | 18 | 104 | 122 |
| 12 | 4 | 37 | 7910 | 17 | 382 | 686 | 1161 |
| 13 | 4 | 39 | 17856 | 19 | 2952 | 1545 | 2463 |
| 14 | 5 | 48 | 14598 | 23 | 1992 | 1425 | 2334 |
| 15 | 5 | 52 | 73677 | 23 | 47656 | 5104 | 10876 |
| 16 | 5 | 64 | 27853 | 26 | 32881 | 20035 | 5970 |

Table B.11: Stable Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 3 | 3 | 0 | 3 | 7 |
| 5 | 2 | 10 | 3 | 5 | 0 | 3 | 5 |
| 6 | 2 | 15 | 5 | 6 | 1 | 6 | 11 |
| 7 | 2 | 21 | 13 | 7 | 2 | 10 | 10 |
| 8 | 2 | 30 | 17 | 10 | 1 | 9 | 12 |
| 9 | 2 | 37 | 75 | 13 | 4 | 22 18 | |
| 10 | 2 | 46 | 2251 | 14 | 75 | 181 | 233 |
| 11 | 2 | 54 | 99 | 16 | 5 | 28 | 24 |
| 12 | 3 | 62 | 797 | 19 | 39 | 93 | 144 |
| 13 | 3 | 77 | 1825 | 22 | 291 | 176 | 337 |
| 14 | 3 | 92 | 819 | 24 | 37 | 102 | 142 |
| 15 | 3 | 104 | 5574 | 26 | 1382 | 584 | 905 |
| 16 | 3 | 117 | 2726 | 29 | 189 | 328 | 645 |

Table B.12: Stable Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 1 | 3 | 0 | 2 | 2 |
| 5 | 1 | 16 | 3 | 5 | 0 | 4 | 4 |
| 6 | 2 | 22 | 5 | 7 | 1 | 6 | 4 |
| 7 | 1 | 32 | 4 | 7 | 1 | 5 | 4 |
| 8 | 2 | 41 | 15 | 10 | 1 | 11 | 10 |
| 9 | 1 | 57 | 4 | 10 | 1 | 8 | 6 |
| 10 | 2 | 68 | 6 | 12 | 1 | 4 | 9 |
| 11 | 1 | 85 | 12 | 13 | 2 | 6 | 12 |
| 12 | 2 | 99 | 14 | 17 | 2 | 11 | 18 |
| 13 | 2 | 117 | 39 | 19 | 3 | 23 | 12 |
| 14 | 2 | 134 | 43 | 21 | 3 | 22 | 20 |
| 15 | 2 | 156 | 324 | 25 | 15 | 86 | 93 |
| 16 | 2 | 177 | 346 | 26 | 15 | 96 | 112 |

# Data sets for the Approximate Algorithm

Table B.13: Grounded Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 2 | 0 | 0 | 0 |
| 5 | 3 | 3 | 1 | 2 | 0 | 1 | 0 |
| 6 | 3 | 6 | 1 | 4 | 0 | 0 | 0 |
| 7 | 2 | 10 | 1 | 5 | 0 | 0 | 1 |
| 8 | 2 | 12 | 1 | 4 | 0 | 0 | 1 |
| 9 | 4 | 16 | 1 | 9 | 0 | 0 | 0 |
| 10 | 3 | 22 | 1 | 9 | 0 | 0 | 0 |
| 11 | 2 | 27 | 1 | 5 | 0 | 1 | 1 |
| 12 | 3 | 30 | 1 | 8 | 0 | 0 | 1 |

Table B.14: Grounded Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 5 | 1 | 3 | 0 | 1 | 0 |
| 5 | 2 | 8 | 1 | 3 | 0 | 0 | 1 |
| 6 | 2 | 13 | 1 | 6 | 0 | 0 | 0 |
| 7 | 2 | 19 | 1 | 7 | 0 | 0 | 0 |
| 8 | 2 | 25 | 1 | 7 | 0 | 1 | 0 |
| 9 | 2 | 29 | 1 | 7 | 0 | 0 | 0 |
| 10 | 2 | 41 | 1 | 9 | 0 | 1 | 1 |
| 11 | 2 | 52 | 1 | 8 | 0 | 1 | 1 |
| 12 | 1 | 60 | 1 | 7 | 0 | 1 | 0 |

Table B.15: Grounded Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 1 | 3 | 0 | 0 | 0 |
| 5 | 1 | 13 | 1 | 4 | 0 | 0 | 0 |
| 6 | 2 | 19 | 1 | 6 | 0 | 0 | 0 |
| 7 | 2 | 28 | 1 | 8 | 0 | 0 | 0 |
| 8 | 2 | 36 | 1 | 9 | 0 | 0 | 1 |
| 9 | 2 | 49 | 1 | 10 | 0 | 1 | 0 |
| 10 | 2 | 61 | 1 | 12 | 1 | 0 | 1 |
| 11 | 1 | 75 | 1 | 11 | 0 | 1 | 1 |
| 12 | 1 | 87 | 1 | 9 | 0 | 1 | 0 |

Table B.16: Preferred Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 4 | 1 | 2 | 0 | 2 | 2 |
| 5 | 2 | 7 | 1 | 4 | 1 | 1 | 3 |
| 6 | 3 | 8 | 1 | 5 | 0 | 1 | 2 |
| 7 | 3 | 12 | 1 | 7 | 1 | 5 | 2 |
| 8 | 3 | 15 | 1 | 8 | 0 | 1 | 1 |
| 9 | 3 | 20 | 1 | 10 | 0 | 2 | 2 |
| 10 | 4 | 21 | 1 | 12 | 0 | 2 | 1 |
| 11 | 4 | 27 | 1 | 14 | 0 | 1 | 0 |
| 12 | 4 | 32 | 1 | 15 | 1 | 2 | 1 |
| 13 | 4 | 41 | 1 | 18 | 1 | 2 | 1 |
| 14 | 4 | 49 | 1 | 21 | 0 | 2 | 1 |
| 15 | 4 | 52 | 1 | 24 | 0 | 2 | 1 |
| 16 | 5 | 60 | 1 | 26 | 1 | 2 | 2 |

Table B.17: Preferred Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 1 | 3 | 0 | 2 | 2 |
| 5 | 2 | 10 | 1 | 5 | 0 | 2 | 1 |
| 6 | 2 | 15 | 1 | 7 | 0 | 1 | 1 |
| 7 | 2 | 22 | 1 | 9 | 0 | 1 | 1 |
| 8 | 2 | 28 | 1 | 11 | 0 | 3 | 1 |
| 9 | 2 | 38 | 1 | 12 | 1 | 3 | 1 |
| 10 | 3 | 45 | 1 | 16 | 1 | 2 | 3 |
| 11 | 2 | 56 | 1 | 17 | 1 | 3 | 1 |
| 12 | 2 | 66 | 1 | 17 | 0 | 2 | 1 |
| 13 | 2 | 81 | 1 | 20 | 1 | 5 | 1 |
| 14 | 3 | 90 | 1 | 25 | 1 | 2 | 2 |
| 15 | 3 | 105 | 1 | 27 | 1 | 2 | 1 |
| 16 | 2 | 122 | 1 | 26 | 0 | 3 | 1 |

Table B.18: Preferred Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 1 | 9 | 1 | 3 | 0 | 1 | 2 |
| 5 | 1 | 15 | 1 | 5 | 0 | 2 | 1 |
| 6 | 1 | 23 | 1 | 6 | 0 | 3 | 2 |
| 7 | 1 | 32 | 1 | 7 | 0 | 1 | 1 |
| 8 | 1 | 42 | 1 | 9 | 0 | 1 | 2 |
| 9 | 2 | 54 | 1 | 11 | 0 | 2 | 1 |
| 10 | 2 | 67 | 1 | 12 | 0 | 1 | 1 |
| 11 | 2 | 81 | 1 | 16 | 0 | 2 | 2 |
| 12 | 2 | 99 | 1 | 16 | 1 | 2 | 1 |
| 13 | 2 | 115 | 1 | 21 | 1 | 1 | 2 |
| 14 | 2 | 134 | 1 | 21 | 0 | 1 | 1 |
| 15 | 2 | 157 | 1 | 26 | 1 | 2 | 2 |
| 16 | 2 | 184 | 1 | 28 | 1 | 2 | 3 |

Table B.19: Stable Extension with Attack Probability 0.25

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 4 | 1 | 2 | 0 | 2 | 2 |
| 5 | 3 | 6 | 1 | 4 | 0 | 2 | 1 |
| 6 | 3 | 10 | 1 | 6 | 1 | 1 | 1 |
| 7 | 3 | 11 | 1 | 7 | 0 | 1 | 1 |
| 8 | 4 | 14 | 1 | 9 | 0 | 1 | 0 |
| 9 | 4 | 19 | 1 | 11 | 0 | 0 | 0 |
| 10 | 4 | 22 | 1 | 13 | 1 | 1 | 1 |
| 11 | 4 | 29 | 1 | 14 | 1 | 1 | 0 |
| 12 | 4 | 37 | 1 | 17 | 0 | 2 | 3 |
| 13 | 4 | 39 | 1 | 19 | 0 | 2 | 2 |
| 14 | 5 | 48 | 1 | 23 | 0 | 1 | 1 |
| 15 | 5 | 52 | 1 | 23 | 1 | 2 | 1 |
| 16 | 5 | 64 | 1 | 26 | 1 | 2 | 1 |

Table B.20: Stable Extension with Attack Probability 0.50

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 1 | 3 | 0 | 1 | 3 |
| 5 | 2 | 10 | 1 | 5 | 0 | 2 | 2 |
| 6 | 2 | 15 | 1 | 6 | 1 | 2 | 1 |
| 7 | 2 | 21 | 1 | 7 | 0 | 2 | 1 |
| 8 | 2 | 30 | 1 | 10 | 0 | 4 | 1 |
| 9 | 2 | 37 | 1 | 13 | 1 | 1 | 1 |
| 10 | 2 | 46 | 1 | 14 | 0 | 2 | 1 |
| 11 | 2 | 54 | 1 | 16 | 1 | 3 | 1 |
| 12 | 3 | 62 | 1 | 19 | 1 | 2 | 2 |
| 13 | 3 | 77 | 1 | 22 | 1 | 2 | 1 |
| 14 | 3 | 92 | 1 | 24 | 1 | 3 | 1 |
| 15 | 3 | 104 | 1 | 26 | 0 | 3 | 1 |
| 16 | 3 | 117 | 1 | 29 | 1 | 2 | 2 |

Table B.21: Stable Extension with Attack Probability 0.75

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 1 | 3 | 0 | 3 | 1 |
| 5 | 1 | 16 | 1 | 5 | 0 | 2 | 1 |
| 6 | 2 | 22 | 1 | 7 | 0 | 3 | 3 |
| 7 | 1 | 32 | 1 | 7 | 0 | 1 | 2 |
| 8 | 2 | 41 | 1 | 10 | 1 | 1 | 1 |
| 9 | 1 | 57 | 1 | 10 | 0 | 4 | 2 |
| 10 | 2 | 68 | 1 | 12 | 1 | 3 | 2 |
| 11 | 1 | 85 | 1 | 13 | 1 | 3 | 3 |
| 12 | 2 | 99 | 1 | 17 | 0 | 1 | 2 |
| 13 | 2 | 117 | 1 | 19 | 0 | 2 | 1 |
| 14 | 2 | 134 | 1 | 21 | 0 | 1 | 1 |
| 15 | 2 | 156 | 1 | 25 | 0 | 2 | 2 |
| 16 | 2 | 177 | 1 | 26 | 0 | 3 | 4 |

# Data sets for the Approximate Algorithm for larger AAF Sizes

Table B.22: Grounded Extension with Attack Probability 0.25 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 4 | 1 | 3 | 0 | 1 | 1 |
| 10 | 4 | 19 | 1 | 9 | 0 | 1 | 1 |
| 15 | 2 | 49 | 1 | 6 | 0 | 2 | 2 |
| 20 | 2 | 89 | 1 | 12 | 0 | 1 | 1 |
| 25 | 1 | 149 | 1 | 8 | 0 | 1 | 1 |
| 30 | 1 | 207 | 1 | 10 | 0 | 1 | 0 |
| 35 | 1 | 293 | 1 | 9 | 0 | 1 | 0 |
| 40 | 1 | 380 | 1 | 10 | 0 | 1 | 0 |
| 45 | 1 | 492 | 1 | 11 | 0 | 1 | 0 |
| 50 | 1 | 602 | 1 | 12 | 0 | 1 | 0 |
| 55 | 1 | 747 | 1 | 15 | 0 | 1 | 3 |
| 60 | 1 | 867 | 1 | 16 | 0 | 1 | 1 |

Table B.23: Preferred Extension with Attack Probability 0.25 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 5 | 1 | 4 | 0 | 2 | 3 |
| 10 | 4 | 23 | 1 | 12 | 0 | 3 | 1 |
| 15 | 4 | 55 | 1 | 22 | 0 | 3 | 2 |
| 20 | 4 | 92 | 1 | 31 | 1 | 4 | 4 |
| 25 | 6 | 153 | 1 | 51 | 1 | 6 | 5 |
| 30 | 6 | 225 | 1 | 67 | 1 | 6 | 4 |
| 35 | 6 | 298 | 1 | 86 | 2 | 7 | 5 |
| 40 | 7 | 394 | 1 | 101 | 2 | 5 | 5 |
| 45 | 7 | 500 | 1 | 120 | 2 | 7 | 5 |
| 50 | 8 | 606 | 1 | 146 | 2 | 6 | 6 |
| 55 | 7 | 733 | 1 | 156 | 4 | 7 | 7 |
| 60 | 8 | 879 | 1 | 198 | 3 | 8 | 6 |

Table B.24: Preferred Extension with Attack Probability 0.50 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 11 | 1 | 5 | 0 | 2 | 1 |
| 10 | 3 | 45 | 1 | 16 | 1 | 5 | 3 |
| 15 | 3 | 104 | 1 | 27 | 1 | 5 | 2 |
| 20 | 3 | 190 | 1 | 37 | 1 | 5 | 3 |
| 25 | 3 | 302 | 1 | 52 | 2 | 5 | 4 |
| 30 | 4 | 438 | 1 | 73 | 2 | 7 | 5 |
| 35 | 4 | 592 | 1 | 91 | 2 | 7 | 6 |
| 40 | 4 | 785 | 1 | 110 | 2 | 9 | 7 |
| 45 | 4 | 984 | 1 | 123 | 2 | 8 | 6 |
| 50 | 4 | 1229 | 1 | 141 | 2 | 8 | 7 |
| 55 | 4 | 1488 | 1 | 154 | 2 | 8 | 6 |
| 60 | 4 | 1774 | 1 | 179 | 2 | 8 | 7 |

Table B.25: Preferred Extension with Attack Probability 0.75 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 15 | 1 | 5 | 0 | 2 | 3 |
| 10 | 2 | 70 | 1 | 15 | 0 | 2 | 3 |
| 15 | 2 | 157 | 1 | 24 | 0 | 5 | 7 |
| 20 | 2 | 288 | 1 | 34 | 0 | 4 | 5 |
| 25 | 2 | 453 | 1 | 46 | 1 | 6 | 6 |
| 30 | 2 | 651 | 1 | 53 | 2 | 7 | 6 |
| 35 | 2 | 893 | 1 | 63 | 2 | 9 | 4 |
| 40 | 2 | 1165 | 1 | 82 | 2 | 12 | 5 |
| 45 | 3 | 1478 | 1 | 100 | 2 | 12 | 6 |
| 50 | 2 | 1847 | 1 | 99 | 2 | 9 | 7 |
| 55 | 2 | 2227 | 1 | 114 | 2 | 10 | 8 |
| 60 | 3 | 2640 | 1 | 135 | 2 | 10 | 7 |

Table B.26: Stable Extension with Attack Probability 0.25 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 6 | 1 | 4 | 0 | 2 | 2 |
| 10 | 4 | 24 | 1 | 13 | 0 | 2 | 1 |
| 15 | 5 | 50 | 1 | 24 | 0 | 1 | 3 |
| 20 | 5 | 93 | 1 | 39 | 1 | 4 | 3 |
| 25 | 6 | 148 | 1 | 52 | 1 | 5 | 4 |
| 30 | 6 | 215 | 1 | 69 | 2 | 6 | 5 |
| 35 | 6 | 300 | 1 | 87 | 2 | 5 | 5 |
| 40 | 6 | 388 | 1 | 93 | 2 | 8 | 4 |
| 45 | 7 | 497 | 1 | 125 | 2 | 5 | 6 |
| 50 | 7 | 627 | 1 | 140 | 2 | 6 | 6 |
| 55 | 8 | 740 | 1 | 165 | 3 | 8 | 7 |
| 60 | 8 | 873 | 1 | 179 | 2 | 6 | 6 |

Table B.27: Stable Extension with Attack Probability 0.50 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 10 | 1 | 5 | 0 | 1 | 4 |
| 10 | 2 | 46 | 1 | 15 | 1 | 3 | 1 |
| 15 | 3 | 105 | 1 | 26 | 1 | 3 | 2 |
| 20 | 3 | 193 | 1 | 42 | 1 | 6 | 4 |
| 25 | 3 | 302 | 1 | 51 | 2 | 7 | 4 |
| 30 | 4 | 442 | 1 | 72 | 2 | 8 | 4 |
| 35 | 4 | 595 | 1 | 88 | 2 | 8 | 6 |
| 40 | 4 | 777 | 1 | 103 | 2 | 8 | 6 |
| 45 | 4 | 996 | 1 | 126 | 2 | 6 | 6 |
| 50 | 4 | 1231 | 1 | 140 | 2 | 10 | 6 |
| 55 | 4 | 1498 | 1 | 160 | 2 | 8 | 7 |
| 60 | 4 | 1762 | 1 | 182 | 2 | 8 | 7 |

Table B.28: Stable Extension with Attack Probability 0.75 for larger AAF Sizes

| AAF Size | Ext Size | Attacks | Preference Sets | Preferences | CTime (ms) | VTime1 (ms) | VTime2 (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 1 | 16 | 1 | 4 | 0 | 2 | 2 |
| 10 | 1 | 71 | 1 | 11 | 0 | 2 | 3 |
| 15 | 2 | 156 | 1 | 25 | 0 | 3 | 5 |
| 20 | 2 | 289 | 1 | 33 | 1 | 5 | 4 |
| 25 | 2 | 445 | 1 | 41 | 1 | 5 | 3 |
| 30 | 2 | 651 | 1 | 53 | 1 | 7 | 6 |
| 35 | 2 | 890 | 1 | 66 | 2 | 10 | 5 |
| 40 | 2 | 1166 | 1 | 79 | 2 | 10 | 7 |
| 45 | 2 | 1480 | 1 | 93 | 2 | 10 | 6 |
| 50 | 2 | 1832 | 1 | 100 | 2 | 9 | 6 |
| 55 | 2 | 2228 | 1 | 119 | 2 | 10 | 7 |
| 60 | 2 | 2656 | 1 | 130 | 2 | 8 | 8 |