An Efficient Algorithm for Computing the Set of Semi-stable Extensions

Gianvincenzo Alfano(✉)

Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria, Rende, Italy
g.alfano@dimes.unical.it

Abstract. Argumentation is one of the most relevant fields in the sphere of Artificial Intelligence. In particular, Dung's abstract argumentation framework (AF) has received much attention in the last twenty years, and many computational issues have been investigated for different argumentation semantics. Specifically, enumerating the sets of arguments prescribed by an argumentation semantics (i.e., *extensions*) is arguably one of the most challenging problems for AFs, and this is the case also for the well-known *semi-stable* semantics.

In this paper, we propose an algorithm for efficiently computing the set of semi-stable extensions of a given AF. Our technique relies on exploiting the computation of grounded extension to snip some arguments in order to obtain a smaller framework (called cut-AF) over which state-of-the-art solvers for enumerating the semi-stable extensions are called, as needed to return the extensions of the input AF.

We experimentally evaluated our technique and found that our approach is orders of magnitude faster than the computation over the whole AF.

Keywords: Abstract argumentation · Semi-stable semantics · Enumeration of semi-stable extensions

1 Introduction

Abstract argumentation has emerged as one of the major fields in Artificial Intelligence [10,15,41,44].

The capability to handle incompatible and conflicting information make argumentation applicable to several real-world scenarios as, for example, building arguments by retrieving information from relational databases [20] (within different context applications), such that a query corresponds to determine from a skeptical viewpoint the truth of such arguments.

In particular, abstract argumentation frameworks (AFs) [21] are a simple, yet powerful formalism for modelling disputes between two or more agents. The formal meaning of an AF is given in terms of argumentation semantics, which intuitively tell us the sets of arguments (called *extensions*) that can collectively be used to support a point of view in a discussion.

A. Cuzzocrea et al. (Eds.): FQAS 2019, LNAI 11529, pp. 139–151, 2019.
https://doi.org/10.1007/978-3-030-27629-4_15

Although the idea underlying AFs is very simple and intuitive, most of the argumentation semantics proposed so far suffer from a high computational complexity [22,24,26–28]. In particular, the enumeration problem of AFs (i.e., the problem of computing all extensions according to some semantics) is intractable for several argumentation semantics [23,35], including the *semi-stable* semantics, one of the more recent semantics introduced in [18] to avoid a problem the stable semantics has. In fact, although stable semantics is one of the oldest way to determine which argument can be accepted [30], it is not always true that, given an argumentation framework, a stable extension for it exists. Complexity bounds and evaluation algorithms for AFs have been deeply investigated in the literature, and the International Competition on Computational Models of Argumentation (ICCMA)[1] has been established for promoting research and development of efficient algorithms for computational models of AFs. A challenging computational tasks of ICCMA is EE-`sst`, that is, enumerating all the extensions of a given AF under the semi-stable semantics.

In this paper, we propose an approach for scaling up the computation of the EE-`sst` problem.

Contributions. The main contributions of the paper are as follows:

- We propose the concept of *cut-AF* that allows us to compute all the semi-stable extensions by focusing only on a smaller portion of the initial AF. Particularly, the cut-AF is built by removing from the whole AF all those relationships and arguments belonging to the grounded extension, which is a proper set of arguments contained in every semi-stable extensions [18].
- We come up with an efficient algorithm for computing the set of all semi-stable extensions. The algorithm enables the computation of the semi-stable extensions by focusing only on the cut-AFs and using state-of-the-art AF solvers.
- An experimental analysis to show the relevance of our approach is presented. It is carried out by comparing our technique with other state-of-the-art solvers able to solve both the enumeration problem of semi-stable semantics and computation of grounded semantics, and show that our technique is at least 400 times faster than the computation from scratch.

2 Preliminaries

We assume the existence of a set *Arg* of *arguments*. An *(abstract) argumentation framework* [21] (*AF*) is a pair $\langle A, \Sigma \rangle$, where $A \subseteq Arg$ is a finite set of *arguments*, and $\Sigma \subseteq A \times A$ is a binary relation over A whose elements are called *attacks*. Thus, an AF can be viewed as a directed graph where nodes correspond to arguments and edges correspond to attacks.

Example 1 (Running example). The pair $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$ where $A_0 = \langle \{a, b, c, d, e, f, g, h\}$ and $\Sigma_0 = \{(a, b), (b, c), (c, d), (d, a), (f, e), (g, h), (e, a), (h, a)\}\rangle$ is an AF, and the corresponding graph is shown in Fig. 1(a).

[1] http://argumentationcompetition.org.

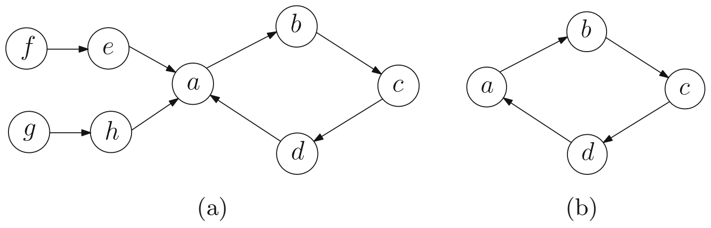


Fig. 1. (a) AF $\mathcal{A}_0$, (b) AF *Cut*($\mathcal{A}_0$).

Given an AF $\langle A, \Sigma\rangle$ and arguments $a, b \in A$, we say that *a attacks b* iff $(a, b) \in \Sigma$, and that a set $S \subseteq A$ attacks b iff there is $a \in S$ attacking b. We use $S^+ = \{b \mid \exists\, a \in S : (a, b) \in \Sigma\}$ to denote the set of arguments attacked by S. For instance, in our running example, we have that $\{b, d\}^+ = \{c, a\}$.

Moreover, we say that $S \subseteq A$ *defends a* iff $\forall b \in A$ such that b *attacks* a, there is $c \in S$ such that c *attacks* b. In our running example, we have that $\{b, d\}$ defends both b and d, as b defends d from the attack of c, and d defends b from the attack of a.

A set $S \subseteq A$ of arguments is said to be:

(i) *conflict-free*, if there are no $a, b \in S$ such that a *attacks* b;
(ii) *admissible*, if it is conflict-free and it defends all its arguments.

For instance, in our example, $\{b, d\}$ is conflict-free and thus it is admissible since, as said earlier, it defends all of its arguments.

An argumentation semantics specifies the criteria for identifying a set of arguments that can be considered "reasonable" together, called *extension.*

A *complete extension* (co) is an admissible set that contains all the arguments that it defends. It is easy to see that, the complete extensions of the AF in Fig. 1(a) are $\{f, g\}, \{a, c, f, g\}, \{b, d, f, g\}$.

A complete extension S is said to be:

– *grounded (gr)* iff it is minimal (w.r.t. $\subseteq$);
– *semi-stable (sst)* iff $S \cup S^+$ is maximal (w.r.t. $\subseteq$).

Given an AF $\mathcal{A}$ and a semantics $\mathcal{S} \in \{\texttt{co}, \texttt{gr}, \texttt{sst}\}$, we use $\mathcal{E}_{\mathcal{S}}(\mathcal{A})$ to denote the set of $\mathcal{S}$-extensions for $\mathcal{A}$, i.e., the set of extensions for $\mathcal{A}$ prescribed by semantics $\mathcal{S}$.

All the above-mentioned semantics admit at least one extension, while the grounded admits exactly one extension [21]. The grounded semantics is called *deterministic* or *unique status* as $|\mathcal{E}_{\texttt{gr}}(\mathcal{A})| = 1$, whereas the semi-stable semantics is called *nondeterministic* or *multiple status* since $|\mathcal{E}_{\texttt{sst}}(\mathcal{A})| \geq 1$.

Example 2. Continuing with our example, we have that the grounded extension of $\mathcal{A}_0$ is $E_{gr} = \{f, g\}$ (i.e., $\mathcal{E}_{\texttt{gr}}(\mathcal{A}_0) = \{\{f, g\}\}$). Moreover, the set of semi-stable extensions is $\mathcal{E}_{\texttt{sst}}(\mathcal{A}_0) = \{\{a, c, f, g\}, \{b, d, f, g\}\}$.

It is well-known that, for any AF $\mathcal{A}$ and semantics $\mathcal{S} \in \{\texttt{gr}, \texttt{sst}\}$, it is the case that $\mathcal{E}_{\mathcal{S}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{co}}(\mathcal{A})$, and let E_{gr} and E_{sst} be the grounded and semi-stable extensions, for every $E \in \mathcal{E}_{\texttt{sst}}(\mathcal{A})$, it holds that $E_{gr} \subseteq E$. Indeed, in the example above, we have that $E_{gr} = \{f, g\} \subseteq E_{sst} = \{a, c, f, g\} \subseteq E_{co} = \{a, c, f, g\}$ and $E_{gr} = \{f, g\} \subseteq E_{sst} = \{b, d, f, g\} \subseteq E_{co} = \{b, d, f, g\}$.

3 Enumerating Semi-stable Extensions

In this section, we provide an approach for efficiently enumerating all the semi-stable extensions of a given AF. Our approach relies on first computing the grounded extension and then using it to define a smaller AF, called *cut-AF*, to be used as the starting point for enumerating the semi-stable extensions.

Definition 1. *Let $\mathcal{A} = \langle A, \Sigma \rangle$ be an AF, and E_{gr} the grounded extension for $\mathcal{A}$. The cut-AF for $\mathcal{A}$ is $Cut(\mathcal{A}) = \langle A_{cut}, \Sigma_{cut} \rangle$ where:*

- $A_{cut} = A \setminus (E_{gr} \cup E_{gr}^{+})$;
- $\Sigma_{cut} = \Sigma \setminus \{(a, b) \mid a \in (E_{gr} \cup E_{gr}^{+}) \text{ or } b \in (E_{gr} \cup E_{gr}^{+})\}$.

Thus, the cut-AF is obtained by removing from the initial AF all the arguments belonging to the grounded extension as well as the arguments attacked by some argument in the grounded extension. Consistently with this, all the attacks towards or from the arguments removed are deleted as well.

Example 3. Continuing with our example, since $E_{gr} = \{f, g\}$, we have that $Cut(\mathcal{A}_0) = \langle A_{cut}, \Sigma_{cut} \rangle$ where:

- $A_{cut} = A_0 \setminus (\{f, g\} \cup \{h, e\}) = \{a, b, c, d\}$, and
- $\Sigma_{cut} = \Sigma_0 \setminus \{(f, e), (e, a), (g, h), (h, a), \} = \{(a, b), (b, c), (c, d), (d, a)\}$.

The graph corresponding to the cut-AF is shown in Fig. 1(b).

Observe that computing the cut-AF con be accomplished in polynomial time w.r.t. the size (i.e., number of arguments/attacks) of the initial AF.

The following theorem states that every semi-stable extension E of an AF $\mathcal{A}$ one-to-one corresponds to a semi-stable extension of the AF $Cut(\mathcal{A})$, and we can obtain a semi-stable extension of the whole AF by joining a semi-stable extension of the cut-AF with the grounded extension of $\mathcal{A}$.

Theorem 1. *Let $\mathcal{A} = \langle A, \Sigma \rangle$ be an AF, E_{gr} the grounded extension for $\mathcal{A}$, and $Cut(\mathcal{A}) = \langle A_{cut}, \Sigma_{cut} \rangle$ the cut-AF for $\mathcal{A}$. Then, $E \in \mathcal{E}_{sst}(\mathcal{A})$ iff $E = E_{gr} \cup E_{cut}$ where $E_{cut} \in \mathcal{E}_{sst}(Cut(\mathcal{A}))$.*

Example 4. Continuing from Example 3, the set of semi-stable extensions of the cut-AF is $\mathcal{E}_{\texttt{sst}}(Cut(\mathcal{A})) = \{\{a, c\}, \{b, d\}\}$. Using the result of Theorem 1, we obtain that $\mathcal{E}_{\texttt{sst}}(\mathcal{A}) = \{\{a, c\} \cup E_{gr}, \{b, d\} \cup E_{gr}\}$, where $E_{gr} = \{f, g\}$. Thus, we obtain the semi-stable extensions $\{f, g, a, c\}$ and $\{f, g, b, d\}$ (c.f. Example 2).

3.1 Algorithm

The pseudo-code of our algorithm for computing the set of semi-stable extensions of an AF is shown in Algorithm 1. It takes as input an AF $\mathcal{A}$, and a percentage value p that is a parameter used for deciding when the computation should be carried out by using the cut-AF or not. In fact, in some cases, such as when the grounded extension of the input AF is empty, the overhead of computing the cut-AF does not pay off because it will correspond to be the whole initial framework, and so, computing the semi-stable extensions over the cut-AF would cost the same as computing the extension on the initial AF plus the overhead of computing the cut-AF.

Thus, we use parameter p to decide when computing or not the cut-AF. In particular, if the grounded extension of the given AF is larger than $p\%$ of the number of arguments in the AF, then the cut-AF is computed; otherwise, the semi-stable extensions are directly computed w.r.t. the whole AF from scratch. Here, computing the grounded extension is polynomial-time (while computing the semi-stable extension is hard), and this suggests that the overhead of computing the grounded extension of the input AF is likely to pay off—in Sect. 4 we thoroughly discuss the results of experiments where different values of p are considered, including $p = 0\%$ which means forcing the algorithm to compute the cut-AF in any case.

Algorithm 1 works as follows. It first computes the grounded extension of the given AF $\mathcal{A}$ (Line 1), and then it checks if the size of the grounded extension is bigger than or equal to $p\%$ of the number of the arguments of $\mathcal{A}$ (Line 2). If this holds, the algorithm proceeds by computing the cut-AF (Line 3). Next, an external AF-solver SST-Solver is called for enumerating the set of extensions of the cut-AF (Line 4), from which the extensions of the whole AF are finally computed at Line 5 using the result of Theorem 1. However, if at Line 2 the size of the grounded extension is smaller than $p\%$ of the number of the arguments of $\mathcal{A}$, then the set of extensions of $\mathcal{A}$ is computed from scratch by calling the external solver SST-Solver with input the whole AF (Line 7). Finally, the set of extensions $\mathcal{E}_{\mathsf{sst}}(\mathcal{A})$ computed by using the cut-AF (Lines 3–5) or not (Line 7) is returned.

Example 5. Continuing with our running example, if $p = 0\%$ then the condition at Line 2 trivially holds since $|E_{gr}| \geq 0$ for every AF. Therefore, the cut-AF $\mathcal{A}_{cut} = \mathit{Cut}(\mathcal{A}) = \langle\{a, b, c, d\}, \{(a, b), (b, c), (c, d), (d, a)\}\rangle$ is computed at Line 3. Next, the set of all semi-stable extensions $\mathcal{E}_{\mathsf{sst}}(\mathcal{A}_{cut}) = \{\{a, c\}, \{b, d\}\}$ of the cut-AF is computed (Line 4), and the set of semi-stable extensions of the whole AF is computed at Line 5 by combining the arguments in the grounded extension with those in the semi-stable extensions of the cut-AF. Therefore, the output of the algorithm is obtained as follows: $\mathcal{E}_{\mathsf{sst}}(\mathcal{A}) = \{\{\{f, g\} \cup \{a, c\}\}, \{\{f, g\} \cup \{b, d\}\}\} = \{\{f, g, a, c\}, \{f, g, b, d\}\}$.

Considering now the case that $p = 5\%$, we have again that $|E_{gr}| \geq p \cdot |A|$ (since $2 \geq 0.05 \cdot 8 = 0.4$), and thus the execution of Algorithm 1 is again as above.

Finally, consider the case that $p = 30\%$ for which we have that $|E_{gr}| \not\geq p \cdot |A|$ (since $2 \not\geq 0.3 \cdot 8 = 2.4$). Thus Algorithm 1 directly computes the set

Algorithm 1. CutSST($\mathcal{A}, p$)

```
Input: AF $\mathcal{A} = \langle A, \Sigma \rangle$,
       A percentage value $p$.
Output: Set $\mathcal{E}_{\mathtt{sst}}(\mathcal{A})$ of semi-stable extensions of $\mathcal{A}$.
begin
1: $E_{gr} = \mathsf{GR\text{-}Solver}(\mathcal{A})$
2: if $|E_{gr}| \geq p \cdot |A|$ then
3:     $\mathcal{A}_{cut} = Cut(\mathcal{A})$
4:     $\mathcal{E}_{\mathtt{sst}}(\mathcal{A}_{cut}) = \mathsf{SST\text{-}Solver}(\mathcal{A}_{cut})$
5:     $\mathcal{E}_{\mathtt{sst}}(\mathcal{A}) = \{E \mid E = E_{gr} \cup E_{cut}$, where $E_{cut} \in \mathcal{E}_{\mathtt{sst}}(\mathcal{A}_{cut})\}$
6: else
7:     $\mathcal{E}_{\mathtt{sst}}(\mathcal{A}) = \mathsf{SST\text{-}Solver}(\mathcal{A})$
8: return $\mathcal{E}_{\mathtt{sst}}(\mathcal{A})$
```

$\mathcal{E}_{\mathtt{sst}}(\mathcal{A})$ of semi-stable extensions by calling the solver SST-Solver with input the whole AF (Line 7).

The following theorems states that Algorithm 1 is sound and complete, provided that the external solvers return the correct results.

Theorem 2. *Given an AF $\mathcal{A}$, if GR-Solver and SST-Solver are sound and complete, then Algorithm 1 returns the set $\mathcal{E}_{sst}(\mathcal{A})$ of semi-stable extensions of $\mathcal{A}$.*

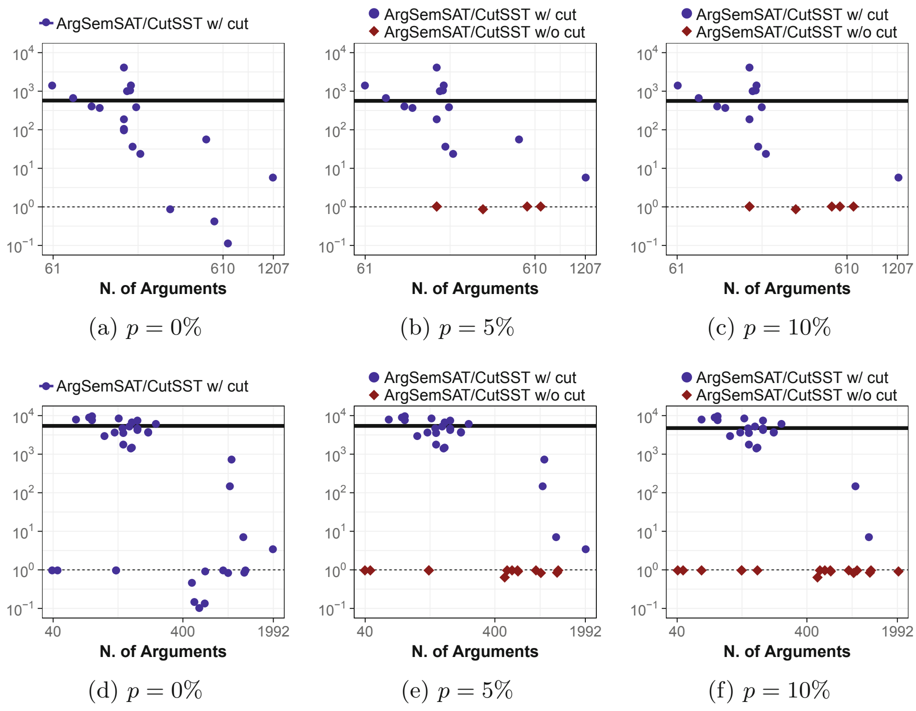
4 Implementation and Experiments

We implemented a C++ prototype to evaluate our technique over benchmark AFs taken from the EE-sst track of ICCMA'17, which consists in determining all the semi-stable extensions of a given AF. Specifically, we used the AFs in the datasets named $E2$ and $E3$ having more than one semi-stable extension.

Particularly, dataset $E2$ (resp. $E3$) consists of 19 (resp. 41) AFs, and a number of arguments contained in AFs of dataset $E2$ (resp. $E3$) that varies from a minimum value of 61 (resp. 40) to a maximum of $1.2K$ (resp. $1.9K$). Furthermore, the range of the number of attacks in the AFs of dataset $E2$ (resp. $E3$) varies from a minimum of 97 (resp. 72) to a maximum of $10.3K$ (resp. $218K$).

Methodology. For every AF $\mathcal{A}$ in each dataset, we first computed the set of all semi-stable extensions of $\mathcal{A}$ by calling Algorithm 1, where as external grounded solver (GR- Solver) is used CoQuiAAS [36], able to resolve the ICCMA'17 track for computing the grounded extension, as well as for computing all the semi-stable extensions (SST-Solver) we used *ArgSemSAT* [19]. Then, the amount of time required by Algorithm 1 was compared with that required by *ArgSemSAT* to compute all semi-stable extension over the given AF $\mathcal{A}$ from scratch.

All experiments were carried out on an Intel Core i7-3770K CPU 3.5 GHz with 12 GB RAM running Ubuntu 16.04.



(a) $p = 0\%$ (b) $p = 5\%$ (c) $p = 10\%$

(d) $p = 0\%$ (e) $p = 5\%$ (f) $p = 10\%$

Fig. 2. Improvement (i.e. the running time of *ArgSemSAT* over the running time of Algorithm 1) for $p = 0\%$ (a and d), $p = 5\%$ (b and e), and $p = 10\%$ (c and f), over the datasets $E2$ (first row) and $E3$ (second row). Circle-shaped data points (coloured blue) represent AFs having a grounded extension larger than or equal to $p\%$ of the number of the arguments, and thus the cut-AF is computed by executing Lines 3–5 of Algorithm 1. Diamond-shaped data points (coloured red) represent AFs having a grounded extension smaller than $p\%$ of the number of arguments, and thus the cut-AF is *not* computed (Line 7 of Algorithm 1 is executed). (Color figure online)

Results. Figure 2 reports the average improvement (log scale) obtained by our algorithm over the computation from scratch for the AFs in the datasets $E2$ (first row), and $E3$ (second row), and for $p = 0\%$ (first column), $p = 5\%$ (second column), $p = 10\%$ (third column).

Specifically, given an AF $\mathcal{A}$ and a percentage value p, we measured the improvement as follows:

$$impr(\mathcal{A}, p) = \frac{\text{running time of } \mathit{ArgSemSAT} \text{ with input } \mathcal{A}}{\text{running time of } \mathit{CutSST}(\mathcal{A}, p)}$$

In Fig. 2, circle-shaped data points (coloured blue) correspond to AFs having a grounded extension larger than or equal to $p\%$ of the number of the arguments. In these cases, the cut-AF is computed by executing Lines 3–5 of Algorithm 1. Diamond-shaped data points (coloured red) represent AFs having a grounded extension smaller than $p\%$ of the number of the arguments, and in this case, both the cut-AF is *not* computed and Line 7 of Algorithm 1 is executed.

For each plot in Fig. 2, a solid black line representing the average improvement obtained for the considered dataset and value of p is reported. Moreover, to easy readability, we also report a dashed grey line corresponding to average improvement equal to 1. Clearly, an improvement strict less than 1 means that the overall overhead of computing the grounded extension, and eventually the cut-AF, does not pay off. However, when the improvement is close to 1, the overhead is negligible.

From the results in Fig. 2, we can draw the following conclusions:

- Our algorithm significantly outperforms the competitor that computes the semi-stable extensions from scratch. In fact, the average improvement is greater than 410 and 2100 over the datasets $E2$ and $E3$, respectively, meaning that Algorithm 1 is on average at least 410 and 2100 times faster than *ArgSemSAT*.
- The smaller the number of arguments of the AFs, the bigger the (average) improvement obtained. In particular, for the datasets $E2$ and $E3$, this implies that the amount of time required decreases from dozens of minutes (computation from scratch) to a few seconds (our algorithm).
- The average improvement remains high for $p = 0\%$, that is, when computing both the grounded extension and the cut-AF irrespectively of the size of the grounded extension. However, the number of AFs for which the improvement is too lower than 1 decreases if $p > 0\%$. In particular, for the datasets $E2$ and $E3$, using $p = 5\%$ is enough for avoiding all the cases for which the improvement is significantly lower than 1. Thus, using p greater than zero allows us to reduce the overhead due to the computation of the grounded extension plus the cut-AF.
- Although increasing the value of p avoids cases where our approach may work worse than the computation from scratch, using too high values of p deteriorates performances on average because the cut-AF is not built even when it would be helpful. In fact, for the datasets $E2$ and $E3$, using $p = 10\%$ entails that the cut-AF is not built in vain for the AFs whose improvements are shown as blue data points in Fig. 2 for $p = 5\%$ and become colored red when passing to $p = 10\%$ (since increasing p entails that cut-AF is no longer computed).
- All in all, the best trade-off between paying the cost of computing the grounded extension along with the cut-AF and risking to have the overhead of the computation of the cut-AF seems to be choosing p greater than zero but no more than 10%.

5 Related Work

Overviews of key concepts in argumentation theory and of formal models of argumentation in the field of Artificial Intelligence can be found in [9,15,16,44].

Further discussion regarding uses of computational argumentation as an Agreement Technology can be found in [42]. Several computational problems of AFs have been studied such as skeptical and credulous reasoning [5], existence of a non-empty extension, and verifying if a set of arguments is an extension under different argumentation semantics [22,24–26]. The complexity of the problem of computing all extensions according to some semantics for AFs has been recently investigated in [35], where it was shown that the enumeration problem is intractable under the semi-stable semantics, and, in particular, it is not in *OutputP* ("output-polynomial time", also known as *TotalP* "total polynomial time" [34]) even for bipartite AFs.

An approach to deal with the problem of enumerating the semi-stable extensions is proposed in [17], where a new algorithm for computing semi-stable semantics using dynamic programming on tree decompositions that runs in linear time on AFs of bounded treewidth is presented. However, this kind of approaches provide advantages only in case of bounded treewidth, noting that that algorithm should not be seen as general solvers that outperform standard techniques on average. This is not case of our technique which performances are not related to the size of treewidth.

Approaches for dividing AFs into subgraphs have been explored also in the context of dynamic AFs—AFs which are updated by adding/removing (set of) attacks/arguments—for which the set of extensions evolves. The division-based method, proposed by [37] and then refined by [12], divides the updated framework into two parts: *affected* and *unaffected*, where only the status of affected arguments is recomputed after updates. Using the results in [37,39] investigated the efficient evaluation of the justification status of a subset of arguments in an AF (instead of the whole set of arguments), and proposed an approach based on answer-set programming for *local* computation. In [38], an AF is decomposed into a set of strongly connected components, yielding sub-AFs located in layers, which are then used for incrementally computing the semantics of the given AF by proceeding layer by layer. Focusing on unique-status semantics, the concept of *influenced set* was introduced in [31–33] to further restrict the set of affected arguments defined in [37], while [2] provided an incremental technique for computing *some* extension of dynamic AFs under multiple-extensions semantics. [13] proposed an approach exploiting the concept of *splitting* of logic programs to deal with dynamic argumentation. [14] investigated whether and how it is possible to modify a given AF so that a desired set of arguments becomes an extension, whereas [43] studied equivalence between two AFs when further information (another AF) is added to both AFs.

Dung's abstract argumentation framework has been extended along several dimensions (e.g. [11,40,45]), and techniques for the efficient computation in dynamic extended AFs were proposed [1,3,4] that, similarly to [32], rely on identifying portions of the AFs that change after performing updates. Finally, the approaches recently proposed in [7,8] focused on the efficient computation of the status of *structured* arguments in dynamic DeLP-programs [29].

The above-cited approaches are related to ours because of the idea of reducing the computation effort by trying to consider a smaller portion of the input AF. However, our technique relies on the novel idea of using cut-AFs through the grounded extension and enabling the modular use of external AF-solvers to compute the set of the semi-stable extensions.

6 Conclusion and Future Work

We introduced a technique for efficiently enumerating the semi-stable extensions of abstract argumentation frameworks. Our approach is modular with respect to the solvers used for computing the grounded extensions, as well as the solver used for the enumeration of semi-stable extensions on the cut-AF—any solver addressing one of these tasks could be plugged-in and exploited for addressing the enumeration problem under the semi-stable semantics. A similar approach is proposed in [6] for enumerating the set of preferred extensions.

We have experimentally investigated the behavior of our technique, and analyzed the conditions under which building the cut-AF is convenient for computing the semi-stable extensions. It turned out that it is worth paying the cost of building the cut-AF after looking at the size of the grounded extension as the computation of the semi-stable extensions over the cut-AF yields significant improvements over the computation from scratch.

Future work will be devoted to extending our technique to the enumeration problem in the presence of other argumentation semantics, such as the *stable* semantics [21]. In fact, a stable extension is a complete extension which attacks all the arguments outside the extension, and the set of stable extensions are a subset of the set of semi-stable extensions; thus, similarly to the semi-stable semantics, the grounded extension is contained in every stable extension. For instance, in our running example the set of the stable extensions coincides with that of the semi-stable extensions, both considering the cut-AF and the whole initial one. However, extending the technique to deal with the stable semantics requires to face up with the fact that a stable extension may not exists for an AF, and checking this is computationally hard [24].

References

1. Alfano, G., Greco, S., Parisi, F.: Computing stable and preferred extensions of dynamic bipolar argumentation frameworks. In: Proceedings of the 1st Workshop on Advances in Argumentation in AI Co-located with AI*IA, pp. 28–42 (2017)
2. Alfano, G., Greco, S., Parisi, F.: Efficient computation of extensions for dynamic abstract argumentation frameworks: an incremental approach. In: Proceedings of IJCAI, pp. 49–55 (2017)
3. Alfano, G., Greco, S., Parisi, F.: Computing extensions of dynamic abstract argumentation frameworks with second-order attacks. In: Proceedings of IDEAS, pp. 183–192 (2018)

4. Alfano, G., Greco, S., Parisi, F.: A meta-argumentation approach for the efficient computation of stable and preferred extensions in dynamic bipolar argumentation frameworks. Intelligenza Artificiale **12**(2), 193–211 (2018)
5. Alfano, G., Greco, S., Parisi, F.: An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks. In: Proceedings of IJCAI (2019, to appear)
6. Alfano, G., Greco, S., Parisi, F.: On scaling the enumeration of the preferred extensions of abstract argumentation frameworks. In: Proceedings of ACM/SIGAPP SAC, pp. 1147–1153 (2019)
7. Alfano, G., Greco, S., Parisi, F., Simari, G.I., Simari, G.R.: An incremental approach to structured argumentation over dynamic knowledge bases. In: Proceeding of KR, pp. 78–87 (2018)
8. Alfano, G., Greco, S., Parisi, F., Simari, G.I., Simari, G.R.: Incremental computation of warranted arguments in dynamic defeasible argumentation: the rule addition case. In: Proceedings of ACM/SIGAPP SAC, pp. 911–917 (2018)
9. Atkinson, K., et al.: Towards artificial argumentation. Artif. Intell. Mag. **38**(3), 25–36 (2017)
10. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowl. Eng. Rev. **26**(4), 365–410 (2011)
11. Baroni, P., Cerutti, F., Giacomin, M., Guida, G.: Encompassing attacks to attacks in abstract argumentation frameworks. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 83–94. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02906-6_9
12. Baroni, P., Giacomin, M., Liao, B.: On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: a division-based method. Artif. Intell. **212**, 104–115 (2014)
13. Baumann, R.: Splitting an Argumentation Framework. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 40–53. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20895-9_6
14. Baumann, R., Brewka, G.: Expanding argumentation frameworks: enforcing and monotonicity results. In: Proceedings of COMMA, pp. 75–86 (2010)
15. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. **171**(10–15), 619–641 (2007)
16. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press, Cambridge (2008)
17. Bliem, B., Hecher, M., Woltran, S.: On efficiently enumerating semi-stable extensions via dynamic programming on tree decompositions. In: Proceedings of COMMA, pp. 107–118 (2016)
18. Caminada, M.: Semi-stable semantics. In: Proceedings of COMMA, pp. 121–130 (2006)
19. Cerutti, F., Giacomin, M., Vallati, M.: ArgSemSAT: solving argumentation problems using SAT. In: Proceedings of COMMA, pp. 455–456 (2014)
20. Deagustini, C.A.D., Dalibón, S.E.F., Gottifredi, S., Falappa, M.A., Chesñevar, C.I., Simari, G.R.: Defeasible argumentation over relational databases. Argument Comput. **8**(1), 35–59 (2017)
21. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
22. Dunne, P.E.: The computational complexity of ideal semantics. Artif. Intell. **173**(18), 1559–1591 (2009)

23. Dunne, P.E., Caminada, M.: Computational complexity of semi-stable semantics in abstract argumentation frameworks. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS, vol. 5293, pp. 153–165. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87803-2_14
24. Dunne, P.E., Wooldridge, M.: Complexity of abstract argumentation. In: Simari, G., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 85–104. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-98197-0_5
25. Dvorák, W., Pichler, R., Woltran, S.: Towards fixed-parameter tractable algorithms for argumentation. In: Proceedings of KR (2010)
26. Dvorák, W., Woltran, S.: Complexity of semi-stable and stage semantics in argumentation frameworks. Inf. Process. Lett. **110**(11), 425–430 (2010)
27. Fazzinga, B., Flesca, S., Parisi, F.: On the complexity of probabilistic abstract argumentation frameworks. ACM Trans. Comput. Log. **16**(3), 22 (2015)
28. Fazzinga, B., Flesca, S., Parisi, F.: On efficiently estimating the probability of extensions in abstract argumentation frameworks. IJAR **69**, 106–132 (2016)
29. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory Pract. Log. Program. (TPLP) **4**(1–2), 95–138 (2004)
30. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Logic Programming, vol. 2, pp. 1070–1080 (1988)
31. Greco, S., Parisi, F.: Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks. In: Proceedings of ECAI, pp. 1668–1669 (2016)
32. Greco, S., Parisi, F.: Incremental computation of deterministic extensions for dynamic argumentation frameworks. In: Michael, L., Kakas, A. (eds.) JELIA 2016. LNCS, vol. 10021, pp. 288–304. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48758-8_19
33. Greco, S., Parisi, F.: Incremental computation of grounded semantics for dynamic abstract argumentation frameworks. In: Aydoğan, R., Baarslag, T., Gerding, E., Jonker, C.M., Julian, V., Sanchez-Anguix, V. (eds.) COREDEMA 2016. LNCS, vol. 10238, pp. 66–81. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57285-7_5
34. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. **27**(3), 119–123 (1988)
35. Kröll, M., Pichler, R., Woltran, S.: On the complexity of enumerating the extensions of abstract argumentation frameworks. In: Proceedings of IJCAI, pp. 1145–1152 (2017)
36. Lagniez, J., Lonca, E., Mailly, J.: CoQuiAAS: a constraint-based quick abstract argumentation solver. In: Proceeding of IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 928–935 (2015)
37. Liao, B.S., Jin, L., Koons, R.C.: Dynamics of argumentation systems: a division-based method. Artif. Intell. **175**(11), 1790–1814 (2011)
38. Liao, B.: Toward incremental computation of argumentation semantics: a decomposition-based approach. Ann. Math. Artif. Intell. **67**(3–4), 319–358 (2013)
39. Liao, B., Huang, H.: Partial semantics of argumentation: basic properties and empirical results. J. Log. Comput. **23**(3), 541–562 (2013)
40. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artif. Intell. **173**(9–10), 901–934 (2009)
41. Modgil, S., Prakken, H.: Revisiting preferences and argumentation. In: Proceedings of IJCAI, pp. 1021–1026 (2011)

42. Modgil, S., et al.: The added value of argumentation: examples and challenges. In: Ossowski, S. (ed.) Agreement Technologies. LGTS, vol. 8, pp. 357–404. Springer, New York (2013). https://doi.org/10.1007/978-94-007-5583-3_21
43. Oikarinen, E., Woltran, S.: Characterizing strong equivalence for argumentation frameworks. Artif. Intell. **175**(14–15), 1985–2009 (2011)
44. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence, 1st edn. Springer, Heidelberg (2009). https://doi.org/10.1007/978-0-387-98197-0
45. Villata, S., Boella, G., Gabbay, D.M., van der Torre, L.W.N.: Modelling defeasible and prioritized support in bipolar argumentation. Ann. Math. Artif. Intell. **66**(1–4), 163–197 (2012)