Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks: An Incremental Approach

Gianvincenzo Alfano, Sergio Greco, Francesco Parisi
Department of Informatics, Modeling, Electronics and System Engineering,
University of Calabria, Italy
{g.alfano, greco, fparisi}@dimes.unical.it

Abstract

Abstract argumentation frameworks (AFs) are a well-known formalism for modelling and deciding many argumentation problems. Computational issues and evaluation algorithms have been deeply investigated for static AFs, whose structure does not change over the time. However, AFs are often dynamic as a consequence of the fact that argumentation is inherently dynamic.

In this paper, we tackle the problem of incrementally computing extensions for dynamic AFs: given an initial extension and an update (or a set of updates), we devise a technique for computing an extension of the updated AF under four well-known semantics (i.e., *complete*, *preferred*, *stable*, and *grounded*). The idea is to identify a reduced (updated) AF sufficient to compute an extension of the whole AF and use state-of-the-art algorithms to recompute an extension of the reduced AF only.

The experiments reveal that, for all semantics considered and using different solvers, the incremental technique is on average two orders of magnitude faster than computing the semantics from scratch.

1 Introduction

Abstract argumentation has emerged as one of the major fields in Artificial Intelligence [Bench-Capon and Dunne, 2007; Rahwan and Simari, 2009; Baroni *et al.*, 2011; Modgil and Prakken, 2011].In particular, abstract argumentation frameworks (AFs) [Dung, 1995] 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 profitably be exploited to support a point of view in a discussion.

Although the idea underlying AFs is very simple and intuitive, most of the argumentation semantics proposed so far suffer from a high computational complexity [Dunne and Wooldridge, 2009]. Complexity bounds and evaluation algorithms for AFs have been deeply investigated in the literature, as also witnessed by the International Competition on Computational Models of Argumentation (ICCMA) [1] whose aim is encouraging research and development of efficient algorithms for computational models of AFs.

However, most research focused on 'static' frameworks, whereas in practice AFs are dynamic systems [Baumann, 2011; Falappa *et al.*, 2011; Liao *et al.*, 2011; Baroni *et al.*, 2014b; Charwat *et al.*, 2015]. In fact, typically an AF represents a temporary situation, and new arguments and attacks can be added/retracted to take into account new available knowledge. For instance, for disputes among users of online social networks [Kökciyan *et al.*, 2016], arguments/attacks are continuously added/retracted by users to express their point of view in response to the last move made by the adversaries (often disclosing as few arguments/attacks as possible).

Surprisingly, the definition of evaluation algorithms and the analysis of the computational complexity taking into account such dynamic aspects have been mostly neglected, whereas in these situations incremental computation techniques could greatly improve performance. In many cases, especially when few updates at a time are performed, the changes made to an AF can result in small changes to the set of its extensions, and recomputing the whole semantics from scratch can be avoided.

With the aim of identifying the portion of an AF affected by an update, the division-based method, proposed by [Liao *et al.*, 2011] and refined by [Baroni *et al.*, 2014b], divides the updated AF into two parts: *affected* and *unaffected*, where only the status of affected arguments needs to be recomputed after updates. Essentially, the set of affected arguments consists of those that are reachable from the updated arguments. Recently, focusing on *unique-extension* semantics, according to which every AF has exactly one extension, in [Greco and Parisi, 2016a; 2016b] we have introduced the concept of *influenced set*, where the initial extension of an AF is used to further restrict the portion of the AF affected by an update. However, the technique in [Greco and Parisi, 2016a; 2016b] only works for the grounded and ideal semantics and does not apply to *multiple-extensions* semantics, which can associate more than one extension to a given AF.

In this paper, we focus on the semantics considered at the last argumentation competition (i.e., the three multiple-extensions semantics *complete*, *preferred*, and *stable*, and the

[1] `http://argumentationcompetition.org`

unique-extension semantics *grounded*) and propose a general approach for incrementally solving the following computational task: Given an AF $\mathcal{A}_0$, an extension for $\mathcal{A}_0$ under semantics $\mathcal{S}$, and an update u, determine an extension of the updated AF $u(\mathcal{A}_0)$ under $\mathcal{S}$. In other words, we explore the possibility of incrementally solving the ICCMA computational task $\mathcal{S}$-SE: Given an AF, determine some $\mathcal{S}$-extension.

Contributions. We introduce an incremental technique for recomputing an extension of an updated AF under four popular semantics. The algorithm consists of three main steps: (*i*) identification of a sub-AF, called *reduced* AF, on the basis of the influenced set and additional information provided by the initial extension (*ii*) using *any* non-incremental algorithm to compute an extension of the reduced AF; and (*iii*) getting the final extension by merging a portion of the initial extension with that computed for the reduced AF.

We performed a thorough experimental analysis showing the effectiveness of our approach for all the semantics considered, even in the case of sets of updates applied simultaneously. Specifically, for each semantics, we compared our technique with the solver that won the ICCMA'15 competition for the computational task $\mathcal{S}$-SE, and show that our technique outperforms the computation from scratch by two orders of magnitude, on average.

2 Preliminaries

We assume the existence of a set Arg of *arguments*. An *(abstract) argumentation framework* [Dung, 1995] (*AF*) is a pair $\langle A, \Sigma\rangle$, where $A \subseteq Arg$ and $\Sigma \subseteq A \times A$ is a binary relation over A whose elements are called *attacks*. Thus, an AF is a directed graph where nodes correspond to arguments and edges correspond to attacks. An argument is an abstract entity whose role is entirely determined by its relationships with other arguments.

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 all arguments that are attacked by S.

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.

A set $S \subseteq A$ 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.

An argumentation semantics specifies the criteria for identifying a set of arguments that can be considered "reasonable" together, called *extension*. A *complete extension* ($\texttt{co}$) is an admissible set that contains all the arguments that it defends. A complete extension S is said to be: *preferred* ($\texttt{pr}$) iff it is maximal (w.r.t. $\subseteq$); *stable* ($\texttt{st}$) iff it attacks each argument in $A \setminus S$; *grounded* ($\texttt{gr}$) iff it is minimal (w.r.t. $\subseteq$).

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

All the above-mentioned semantics except the stable admit at least one extension, and the grounded admits exactly one extension [Dung, 1995]. It is well-known that, for any AF $\mathcal{A}$, $\mathcal{E}_\texttt{gr}(\mathcal{A}) \subseteq \mathcal{E}_\texttt{co}(\mathcal{A})$ and $\mathcal{E}_\texttt{st}(\mathcal{A}) \subseteq \mathcal{E}_\texttt{pr}(\mathcal{A}) \subseteq \mathcal{E}_\texttt{co}(\mathcal{A})$.

Example 1 The set of admissible sets for the AF $\mathcal{A}_0$ shown in Fig.1 (where attack (c, f) does not belong to $\mathcal{A}_0$) is $\{\emptyset, \{b\}, \{g\}, \{a, g\}, \{b, g\}, \{f, g\}, \{a, g, f\}, \{b, g, f\}\}$, and $\mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ with $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$ is as reported in the second column of Table 1. □

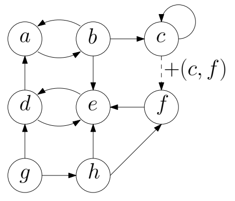

Figure 1: AFs $\mathcal{A}_0$ and $\mathcal{A} = +(c, f)(\mathcal{A}_0)$

$\mathcal{S}$	$\mathcal{E}_\mathcal{S}(\mathcal{A}_0)$	$\mathcal{E}_\mathcal{S}(\mathcal{A}))$
co	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$	$\{\{g\}, \{a, g\}, \{b, f, g\}\}$
pr	$\{\{a, f, g\}, \{b, f, g\}\}$	$\{\{a, g\}, \{b, f, g\}\}$
st	$\{\{b, f, g\}\}$	$\{\{b, f, g\}\}$
gr	$\{\{f, g\}\}$	$\{\{g\}\}$

Table 1: Sets of extensions for $\mathcal{A}_0$ and $\mathcal{A} = +(c, f)(\mathcal{A}_0)$.

The argumentation semantics can be also defined in terms of *labelling* [Baroni *et al.*, 2011]. A labelling for an AF $\mathcal{A} = \langle A, \Sigma\rangle$ is a total function $L : A \to \{\text{IN}, \text{OUT}, \text{UN}\}$ assigning to each argument a label. $L(a) = \text{IN}$ means that argument a is accepted, $L(a) = \text{OUT}$ means that a is rejected, while $L(a) = \text{UN}$ means that a is undecided.

Let $in(L) = \{a \mid a \in A \wedge L(a) = \text{IN}\}$, $out(L) = \{a \mid a \in A \wedge L(a) = \text{OUT}\}$, and $un(L) = \{a \mid a \in A \wedge L(a) = \text{UN}\}$. In the following, we also use the triple $\langle in(L), out(L), un(L)\rangle$ to represent the labelling L.

Given an AF $\mathcal{A} = \langle A, \Sigma\rangle$, a labelling L for $\mathcal{A}$ is said to be *admissible (or legal)* if $\forall a \in in(L) \cup out(L)$ it holds that (*i*) $L(a) = \text{OUT}$ iff $\exists\, b \in A$ such that $(b, a) \in \Sigma$ and $L(b) = \text{IN}$; and (*ii*) $L(a) = \text{IN}$ iff $L(b) = \text{OUT}$ for all $b \in A$ such that $(b, a) \in \Sigma$. Moreover, L is a *complete* labelling iff conditions (*i*) and (*ii*) hold for all $a \in A$.

Between complete extensions and complete labellings there is a bijective mapping defined as follows: for each extension E there is a unique labelling $L = \langle E, E^+, A \setminus (E \cup E^+)\rangle$ and for each labelling L there is a unique extension $in(L)$. We say that L is the labelling *corresponding* to E. For instance, in Example 1, $\langle\{a, f, g\}, \{b, d, e, h\}, \{c\}\rangle$ is the labelling corresponding to the extension $\{a, f, g\}$, while $\langle\{b, f, g\}, \{a, c, d, e, h\}, \emptyset\rangle$ corresponds to $\{b, f, g\}$.

In the following, we say that the *status of an argument* a w.r.t. a labelling L (or its corresponding extension $in(L)$) is IN (resp., OUT, UN) iff $L(a) = \text{IN}$ (resp., $L(a) = \text{OUT}$, $L(a) = \text{UN}$). We will avoid to mention explicitly the labelling (or the extension) whenever it is understood.

Updates. Performing an update on an AF $\mathcal{A}_0$ means modifying it into an AF $\mathcal{A}$ by adding or removing arguments or attacks. In the following, we will present our results by focusing on updates consisting of adding/deleting one attack between arguments of $\mathcal{A}_0$. Then we will show in Section 4.2 that the results for single attack updates extend to the case of multiple updates. In fact, applying a set of updates can be reduced to applying a single update. In our experimental analysis we considered both single and multiple updates.

update $+(a,b)$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN			$\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}$
	UN		$\texttt{co}, \texttt{gr}$	$\texttt{co}, \texttt{pr}, \texttt{gr}$
	OUT	$\texttt{co}, \texttt{pr}, \texttt{st}$	$\texttt{co}, \texttt{gr}$	$\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}$

Table 2: Cases for which $E_0 \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ for $u = +(a, b)$.

update $-(a,b)$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN	NA	NA	
	UN	NA		$\texttt{co}, \texttt{pr}, \texttt{gr}$
	OUT	$\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}$	$\texttt{co}, \texttt{pr}, \texttt{gr}$	$\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}$

Table 3: Cases for which $E_0 \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ for $u = -(a, b)$.

We use $+(a, b)$, with $a, b \in A_0$ and $(a, b) \notin \Sigma_0$, (resp. $-(a, b)$, with $(a, b) \in \Sigma_0$) to denote the addition (resp. deletion) of an attack (a, b), and $u(\mathcal{A}_0)$ to denote the application of update $u = \pm(a, b)$ to AF $\mathcal{A}_0$ (where $\pm$ means either $+$ or $-$). Applying an update u to an AF implies that its semantics (set of extensions or labellings) changes, as shown by Table 1 which reports the sets of extensions for the AFs of Figure 1 before and after performing the update $+(c, f)$.

Concerning the addition (resp. deletion) of a set of isolated arguments, it is easy to see that if $\mathcal{A}$ is obtained from $\mathcal{A}_0$ through the addition (resp. deletion) of a set S of isolated argument, then, let E_0 be an extension for $\mathcal{A}_0$, $E = E_0 \cup S$ (resp. $E = E_0 \setminus S$) is an extension for $\mathcal{A}$ that can be trivially computed. Of course, if arguments in S are not isolated, for addition we can first add isolated arguments and then add attacks involving these arguments, while for deletion we can first delete all attacks involving arguments in S. Thus we do not consider these kinds of update in the following.

3 Arguments Influenced by an Update

In this section, we first provide some sufficient conditions ensuring that a given $\mathcal{S}$-extension for an AF $\mathcal{A}$ continues to be an $\mathcal{S}$-extension for the updated AF $u(\mathcal{A})$. Then, we introduce the *influenced set* which intuitively consists of the set of arguments whose status may change after performing an update.

Updates preserving a given initial extension. Given an update $\pm(a, b)$ and an initial extension E_0 corresponding to L_0, for each pair of initial statuses $L_0(a)$ and $L_0(b)$ of the arguments involved in the update, Tables 2 and 3 tell us the semantics for which E_0 is still an extension after the update.

Proposition 1 *Let $\mathcal{A}_0$ be an AF, $\mathcal{S}$ a semantics, $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ an extension of $\mathcal{A}_0$ under semantics $\mathcal{S}$, L_0 the labelling corresponding to E_0, and u an update. If $\mathcal{S}$ is in the cell $\langle L_0(a), L_0(b)\rangle$ of Table 2 and $u = +(a, b)$ (resp., of Table 3 and $u = -(a, b)$), then $E_0 \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$.*

The results in Tables 2 and 3 concerning $\texttt{gr}$ follow from those in [Boella *et al.*, 2009a; 2009b], where the principles according to which the grounded extension does not change when attacks are added/removed have been studied.

In the following, given an AF $\mathcal{A}_0$ and an $\mathcal{S}$-extension E_0 for it, we say that an update u is *irrelevant* w.r.t. E_0 and $\mathcal{S}$ iff the conditions of Proposition 1 hold. Otherwise, u is *relevant*.

Example 2 Consider $\mathcal{A}_0$ of Figure 1 and its sets of extensions listed in the second column of Table 1. $E_0 = \{b, f, g\}$ is an extension according to semantics $\mathcal{S} \in \{\texttt{co},\texttt{pr},\texttt{st}\}$. Thus, $L_0(c) = \text{OUT}$ and $L_0(f) = \text{IN}$, and using Proposition 1 it follows that for update $u = +(c, f)$ E_0 is still an extension of $u(\mathcal{A}_0)$ (see the last column of Table 1). Thus $+(c, f)$ is irrelevant w.r.t. E_0 and $\mathcal{S}$. However, $+(c, f)$ is relevant w.r.t. $E_0 = \{a, f, g\}$ and all the semantics (in this case $L_0(c) = \text{UN}$ and $L_0(f) = \text{IN}$, and no semantics is listed in the cell $\langle\text{UN}, \text{IN}\rangle$ of Table 2). $\square$

It is important to note that Tables 2 and 3 are not meant to be exhaustive, as more conditions can be found for which an $\mathcal{S}$-extension is preserved after an update. For instance, for the grounded semantics, the initial extension is preserved also if $L_0(a) = \text{OUT}$ and $L_0(b) = \text{IN}$ and argument a of updated $+(a, b)$ is not reachable from b. Here we provided a simple set of conditions that can be easily checked by just looking at the initial labelling L_0. Our technique can be trivially extended by considering a more general set of such conditions.

Influenced set. For irrelevant updates, the influenced set will be empty (in this case, the initial extension will be immediately returned as an extension of the updated AF by our algorithm). If none of the conditions of Proposition 1 holds (i.e., the update is relevant), then the influenced set may turn out to be not empty. In such case, the influenced set will be used to delineate a portion of the argumentation framework, called *reduced AF*, that we will use to recompute (a portion of) an extension for the updated AF.

Given an AF $\mathcal{A} = \langle A, \Sigma\rangle$ and an argument $b \in A$, we use $\mathit{Reach}_\mathcal{A}(b)$ to denote the set of arguments that are reachable from b in the graph $\mathcal{A}$.

Definition 1 (Influenced set) Let $\mathcal{A} = \langle A, \Sigma\rangle$ be an AF, $u = \pm(a, b)$, E an extension of $\mathcal{A}$ under semantics $\mathcal{S}$, and let

– $\mathcal{I}_0(u, \mathcal{A}, E) = \begin{cases} \emptyset \;\; \textit{if } u \textit{ is irrelevant w.r.t. } E \textit{ and } \mathcal{S} \textit{ or} \\ \quad \exists (z, b) \in \Sigma \textit{ s.t. } z \neq a \wedge z \in E \wedge \\ \quad z \notin \mathit{Reach}_\mathcal{A}(b); \\ \{b\} \;\; \textit{otherwise}; \end{cases}$

– $\mathcal{I}_{i+1}(u, \mathcal{A}, E) = \mathcal{I}_i(u, \mathcal{A}, E) \cup \{y \mid \exists (x, y) \in \Sigma \textit{ s.t. } x \in \mathcal{I}_i(u, \mathcal{A}, E) \wedge \nexists (z, y) \in \Sigma \textit{ s.t. } z \in E \wedge z \notin \mathit{Reach}_\mathcal{A}(b)\}$.

The *influenced set* of u w.r.t. $\mathcal{A}$ and E is $\mathcal{I}(u, \mathcal{A}, E) = \mathcal{I}_n(u, \mathcal{A}, E)$ such that $\mathcal{I}_n(u, \mathcal{A}, E) = \mathcal{I}_{n+1}(u, \mathcal{A}, E)$. $\square$

Example 3 Consider the the AF $\mathcal{A}_0 = \langle A_0, \Sigma_0\rangle$ of Figure 1 and the update $u = +(c, f)$. We have that $\mathit{Reach}_{\mathcal{A}_0}(f) = A_0 \setminus \{g, h\}$. The influenced set depends on the initial extension chosen. For the extension $\{b, f, g\}$ of Example 2, we have that the influenced set is empty as u is irrelevant. For the extension $E_0 = \{a, f, g\}$, the influenced set is $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{f, e\}$. Indeed, $d \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ since it is attacked by $g \in E_0$ which is not reachable from f. Thus the arguments that can be reached from d do not belong to $\mathcal{I}(u, \mathcal{A}_0, E_0)$. If we consider the initial grounded extension $\{f, g\}$, then $\{f, e\}$ turns out to be the influenced set again. $\square$

4 Incremental Computation of Extensions

Given the influenced set, we define a subgraph, called *reduced AF*, that will be used to compute the status of the influenced arguments, thus providing an extension that will be combined with that of initial AF to obtain an extension of the updated AF, for every semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$.

For any AF $\mathcal{A} = \langle A, \Sigma \rangle$ and set $S \subseteq A$ of arguments, we denote with $\Pi(S, \mathcal{A}) = \langle S, \Sigma \cap S \times S \rangle$ the subgraph of $\mathcal{A}$ induced by the nodes in S. Moreover, given two AFs $\mathcal{A}_1 = \langle A_1, \Sigma_1 \rangle$ and $\mathcal{A}_2 = \langle A_2, \Sigma_2 \rangle$, we denote as $\mathcal{A}_1 \sqcup \mathcal{A}_2 = \langle A_1 \cup A_1, \Sigma_1 \cup \Sigma_2 \rangle$ the *union* of the two AFs.

Definition 2 (Reduced AF) Let $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$ be an AF, $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ an extension for $\mathcal{A}_0$ under a semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$, and $u = \pm(a, b)$ an update. Let $u(\mathcal{A}_0) = \langle A, \Sigma \rangle$ be the AF updated using u. The reduced AF for $\mathcal{A}_0$ w.r.t. E_0 and u (denoted as $\mathcal{R}(u, \mathcal{A}_0, E_0)$) is as follows.
– $\mathcal{R}(u, \mathcal{A}_0, E_0)$ is empty if $\mathcal{I}(u, \mathcal{A}_0, E_0)$ is empty.
– $\mathcal{R}(u, \mathcal{A}_0, E_0) = \Pi(\mathcal{I}(u, \mathcal{A}_0, E_0), u(\mathcal{A}_0)) \sqcup \mathcal{A}_1 \sqcup \mathcal{A}_2$ where:
  i) $\mathcal{A}_1$ is the union of the AFs $\langle \{a, b\}, \{(a, b)\} \rangle$ s.t. $(a, b) \in \Sigma$, $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$, $a \in E_0$, and $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$;
  ii) $\mathcal{A}_2$ is the union of the AFs $\langle \{c\}, \{(c, c)\} \rangle$ s.t. there is $(e, c) \in \Sigma$, $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$, $e \notin (E_0 \cup E_0^+)$, and $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$. □

Hence, AF $\mathcal{R}(u, \mathcal{A}_0, E_0)$ contains, in addition to the subgraph of $u(\mathcal{A}_0)$ induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$, additional nodes and edges containing needed information on the "external context", i.e. information about the status of arguments which are attacking some argument in $\mathcal{I}(u, \mathcal{A}_0, E_0)$. Using fake arguments/attacks to represent external contexts has been exploited in [Baroni *et al.*, 2014a] where decomposability properties of argumentation semantics are investigated.

Example 4 For our running example, if $E_0 = \{a, f, g\}$ and $u = +(c, f)$, the reduced AF $\mathcal{R}(+(c, f), \mathcal{A}_0, E_0)$ consists of the subgraph induced by $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{f, e\}$ plus the edge (f, f) as there is the attack (c, f) in the updated AF from an uninfluenced argument c labelled as UN toward the influenced argument f. Hence, $\mathcal{R}(+(c, f), \mathcal{A}_0, E_0) = \langle \{e, f\}, \{(f, f), (f, e)\} \rangle$. □

Theorem 1 *Let $\mathcal{A}_0$ be an AF, and $\mathcal{A} = u(\mathcal{A}_0)$ be the AF resulting from performing update $u = \pm(a, b)$ on $\mathcal{A}_0$. Let $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ be an extension for $\mathcal{A}_0$ under a semantics $\mathcal{S} \in \{co, pr, st, gr\}$. Then, if $\mathcal{E}_\mathcal{S}(\mathcal{R}(u, \mathcal{A}_0, E_0))$ is not empty, then there is an extension $E \in \mathcal{E}_\mathcal{S}(\mathcal{A})$ for the updated AF $\mathcal{A}$ such that $E = (E_0 \setminus \mathcal{I}(u, \mathcal{A}_0, E_0)) \cup E_d$ where E_d is an S-extension for reduced AF $\mathcal{R}(u, \mathcal{A}_0, E_0)$.*

Example 5 Continuing our example, for the preferred semantics, let $E_0 = \{a, f, g\}$ and $u = +(c, f)$, we have that $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{f, e\}$, and $\mathcal{R}(+(c, f), \mathcal{A}_0, E_0) = \langle \{e, f\}, \{(f, f), (f, e)\} \rangle$. Thus, using the theorem, there is an extension E of the updated AF such that $E = (\{a, f, g\} \setminus \{f, e\}) \cup E_d$ where $E_d = \emptyset$ is a preferred extension of the reduced AF. In fact, $E = \{a, g\} \in \mathcal{E}_{\texttt{pr}}(u(\mathcal{A}_0))$. □

It is worth noting that the set of extensions of an AF can be empty only for the stable semantics. Thus, in the case that this happens for the reduced AF (i.e., $\mathcal{E}_\mathcal{S}(\mathcal{R}(u, \mathcal{A}_0, E_0)) = \emptyset$), the theorem does not give a method to determine an extension of the updated AF, as shown in the following example.

Algorithm 1 Incr-Alg($\mathcal{A}_0, u, \mathcal{S}, E_0$, Solver$_\mathcal{S}$)

```
Input: AF $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$, update $u = \pm(a, b)$,
       semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$, extension $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$,
       function Solver$_\mathcal{S}(\mathcal{A})$ returning an $\mathcal{S}$-extension for AF $\mathcal{A}$ if it
       exists, $\perp$ otherwise;
Output: An $\mathcal{S}$-extension $E \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ if it exists, $\perp$ otherwise;
1: $S = \mathcal{I}(u, \mathcal{A}_0, E_0)$;
2: if ($S = \emptyset$) then
3:     return $E_0$;
4: $\mathcal{A}_d = \mathcal{R}(u, \mathcal{A}_0, E_0)$;
5: Let $E_d$ = Solver$_\mathcal{S}(\mathcal{A}_d)$;
6: if ($E_d \neq \perp$) then
7:     return $E = (E_0 \setminus S) \cup E_d$;
8: else
9:     return Solver$_\mathcal{S}(u(\mathcal{A}_0))$;
```

Example 6 Let $\mathcal{A}_0 = \langle \{a, b, c, d, e\}, \{(a, b), (b, c), (b, d), (c, d), (c, e), (e, c)\} \rangle$. Its stable extensions are $\{a, c\}$ and $\{a, d, e\}$. For update $u = +(d, d)$, depending on the initial extension, the influenced set is either $\mathcal{I}(u, \mathcal{A}, \{a, c\}) = \emptyset$ (as u is irrelevant w.r.t. $\{a, c\}$ and $\texttt{st}$) or $\mathcal{I}(u, \mathcal{A}, \{a, d, e\}) = \{d\}$. Thus, starting from the extension $\{a, c\}$ we directly know $\{a, c\}$ is a stable extension of the updated AF. However, starting from $\{a, d, e\}$, the reduced AF will be $\mathcal{R}(u, \mathcal{A}_0, \{a, d, e\}) = \langle \{d\}, \{(d, d)\} \rangle$, which has no stable extension. In this case, the theorem does not provide a stable extension of the updated AF, thought a stable extension exists: that obtained by starting from the initial extension $\{a, c\}$.

It is worth noting that, if we consider the preferred semantics, for which the starting extensions are again $\{a, c\}$ and $\{a, d, e\}$, a preferred extension of the updated AF can be obtained no matter what starting extension is chosen. In particular, as the preferred extension for reduced AF $\langle \{d\}, \{(d, d)\} \rangle$ is the empty set, it follows that $(\{a, d, e\} \setminus \{d\}) \cup \emptyset = \{a, d\}$ is a preferred extension of the updated AF. □

4.1 Incremental Algorithm

Our algorithm for computing an extension of an updated AF is shown in Algorithm 1. Besides taking as input an initial AF $\mathcal{A}_0$, an update u, a semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$, and an extension $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$, it also takes as input a function that computes an $\mathcal{S}$-extension for an AF, if any. In particular, function Solver$_\mathcal{S}(\mathcal{A})$ will be used to compute an extension of the reduced AF, which will be then combined with the portion of the initial extension that does not change in order to obtain an extension for the updated AF (as stated in Theorem 1).

More in detail, Algorithm 1 works as follows. First, the influenced set of $\mathcal{A}_0$ w.r.t. update u and the given initial extension E_0 is computed (Line 1). If it is empty, then E_0 will be still an extension of the updated AF under the given semantics $\mathcal{S}$, and thus it is returned (Line 3). Otherwise, the reduced AF $\mathcal{A}_d$ is computed at Line 4, and function Solver$_\mathcal{S}$ is invoked to compute an $\mathcal{S}$-extension of $\mathcal{A}_d$, if any. If $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{gr}\}$, then $\mathcal{A}_d$ will have an extension E_d, which is combined with $E_0 \setminus S$ at Line 7 to get an extension for the updated AF. For the stable semantics, if $\mathcal{E}_{\texttt{st}}(\mathcal{R}(u, \mathcal{A}_0, E_0))$ is not empty, then the algorithm proceeds as for the other semantics (Line 7). Otherwise, function Solver$_\mathcal{S}$ is invoked to compute a stable extension of the whole updated AF $u(\mathcal{A}_0)$, if any.

Theorem 2 *Let $\mathcal{A}_0$ be an AF, $u = \pm(a, b)$, and $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ an extension for $\mathcal{A}_0$ under $\mathcal{S} \in \{co, pr, st, gr\}$. If $\mathsf{Solver}_\mathcal{S}$ is sound and complete then Algorithm 1 computes $E \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ if $\mathcal{E}_\mathcal{S}(u(\mathcal{A}_0)) \neq \emptyset$, otherwise it returns $\perp$.*

4.2 Applying Multiple Updates Simultaneously

We now show how our approach extends to the case of multiple updates. In fact, performing a set of updates $U = \{+(a_1, b_1), \ldots, +(a_n, b_n),\ -(a'_1, b'_1),\ \ldots, -(a'_m, b'_m)\}$ on $\mathcal{A}_0$ can be reduced to performing a single update $+(v, w)$ on an AF whose definition depends on both the set of updates U and the initial $\mathcal{S}$-extension E_0, as detailed in what follows.

Given a set U of updates for an AF $\mathcal{A}_0$, and an $\mathcal{S}$-extension E_0 for $\mathcal{A}_0$, we use U^* to denote the subset of U consisting of the relevant updates (that is, the updates in U for which the conditions of Proposition 1 do not hold).

Definition 3 (AF for applying a set of updates) Let $\mathcal{A} = \langle A, \Sigma\rangle$ be an AF, and E_0 an $\mathcal{S}$-extension for $\mathcal{A}$. Let

- $\Sigma^+ = \{(a_1, b_1), ..., (a_n, b_n)\} \subseteq (A \times A) \setminus \Sigma$, and
- $\Sigma^- = \{(a'_1, b'_1), ..., (a'_m, b'_m)\} \subseteq \Sigma$

such that $\Sigma^+ \cap \Sigma^- = \emptyset$ be two sets of attacks. Let $U = \{+(a_i, b_i) \,|(a_i, b_i) \in \Sigma^+\} \cup \{-(a_j, b_j) \,|(a_j, b_j) \in \Sigma^-\}$ be a set of updates, and $U^* \subseteq U$ be the set of relevant updates w.r.t. E_0 and $\mathcal{S}$. Then, $\mathcal{A}^U_{E_0} = \langle A^U, \Sigma^U\rangle$ denotes the AF obtained from $\mathcal{A}$ as follows:

- $A^U = A \cup \{x_i, y_i \mid +(a_i, b_i) \in U^*\} \cup \{x'_j, y'_j \mid -(a_j, b_j) \in U^*\} \cup \{v, w, w'\}$, where all $x_i, y_i, x'_j, y'_j, w, w'$, and v are new arguments not occurring in A, and
- $\Sigma^U = (\Sigma \setminus \Sigma^-) \cup \{(a_i, b_i) \mid +(a_i, b_i) \in (U \setminus U^*)\} \cup$
  $\{(a_i, x_i), (x_i, y_i), (y_i, b_i) \mid +(a_i, b_i) \in U^*\} \cup$
  $\{(a_j, x'_j), (x'_j, y'_j), (y'_j, b_j) \mid -(a_j, b_j) \in U^*\} \cup$
  $\{(w, y_i) \mid +(a_i, b_i) \in U^*\} \cup$
  $\{(w', y'_j) \mid -(a_j, b_j) \in U^*\} \cup \{(w, w')\}$. $\square$

Theorem 3 *Let $\mathcal{A}_0 = \langle A_0, \Sigma_0\rangle$ be an AF, and U a set of updates. Let $\mathcal{A}$ be the AF obtained from $\mathcal{A}_0$ by performing all updates in U on it. Then, for any semantics $\mathcal{S} \in \{co, pr, st, gr\}$, $E \in \mathcal{E}_\mathcal{S}(\mathcal{A})$ iff there is $E^U \in \mathcal{E}_\mathcal{S}(+(v, w)(\mathcal{A}^U_{E_0}))$ such that $E^U \cap A_0 = E$.*

5 Implementation and Experiments

We implemented a C++ prototype and, for each semantics $\mathcal{S}$, compared the performance of our technique with that of the solver that won the last ICCMA competition for the computational task $\mathcal{S}$-SE: Given an AF, determine some $\mathcal{S}$-extension.

Datasets. We used the ICCMA'15 benchmarks. The benchmark AFs have three different AFs' structures: (i) `TestSetGr` consists of AFs with a very large grounded extension and many arguments in general; (ii) `TestSetSt` consists of AFs with many complete/preferred/stable extensions; and (iii) `TestSetSCC` consists of AFs with a rich structure of strongly connected components. In particular, for each of these test sets, three classes of AFs of different sizes were defined: `Small`, `Medium`, and `Large`. However, `TestSetStLarge` was removed from the competition as the majority of the solvers could not solve any of those AFs. For space limitations, we presents the results obtained for `TestSetGrSmall` and `TestSetGrLarge`, and `TestSetStSmall` and `TestSetStMedium`, but similar results were obtained also for the other datasets.

Methodology. For each semantics $\mathcal{S}$, for each AF $\mathcal{A}_0 = \langle A_0, \Sigma_0\rangle$ in each dataset, we considered every $\mathcal{S}$-extension E_0 of $\mathcal{A}_0$ as an initial extension. Then, when considering single updates, we randomly selected an update of the form $\pm(a, b)$, while for multiple updates we randomly generated a set U of single updates. Next, we computed an $\mathcal{S}$-extension E for the updated AF $u(\mathcal{A}_0)$ by calling Algorithm 1, where, for each semantics $\mathcal{S}$, we used as $\mathsf{Solver}_\mathcal{S}$ the solver that won the ICCMA'15 competition for the task $\mathcal{S}$-SE: *CoQuiAAS* [Lagniez *et al.*, 2015] for $\mathcal{S} = \mathtt{co}$ and $\mathcal{S} = \mathtt{gr}$, *Cegartix* [Dvorák *et al.*, 2014] for $\mathcal{S} = \mathtt{pr}$, and *ASPARTIX-D* [Gaggl and Manthey, 2015] for $\mathcal{S} = \mathtt{st}$. Then, the average run time of our algorithm to compute an $\mathcal{S}$-extension was compared with the average run time of the best ICCMA solver to compute an $\mathcal{S}$-extension for $u(\mathcal{A}_0)$ from scratch.

All experiments have been carried out on an Intel Core i7-3770K CPU 3.5GHz with 12GB RAM running Ubuntu 14.04.

Results. Figure 2 reports the average run times (log scale) of the competitors and *Incr-Alg* for different semantics and datasets. In particular, the graphs also report the run times of *Incr-Alg* for recomputing the extensions after performing sets S of updates simultaneously, with $|S|$ being a percentage of the number of attacks in the initial AFs. The run times of the competitors for these cases were almost equal to their run times for single updates, and are not shown for the sake of readability of the graphs. Besides the data points, for each series, Figure 2 also shows the (solid) lines obtained by linear regression. However, some datasets consist of clusters of AFs that differ substantially on the number of arguments/attacks, and thus in these cases the linear regression line would just connect the centroids of the clusters, becoming not useful. For this reason, we only show the largest cluster for `TestSetStSmall` (the other one consists of only 3 data points), and the two clusters of `TestSetStMedium` separately (the former for $\mathcal{S} = \mathtt{pr}$ and the latter for $\mathcal{S} = \mathtt{st}$).

Moreover, the results obtained for the complete semantics are not shown as they were analogous to those obtained for the grounded semantics. Also, considering updates consisting of adding/removing arguments does not affect the efficiency.

The experiments also showed that, on average, the size of the reduced AF w.r.t. that of the input AF is about 9% for single updates and 52% for multiple updates with about 1% of the attacks updated. Moreover, considering also addition/removal of arguments does not affect the efficiency.

From these results, we can draw the following conclusions:

– Our algorithm significantly outperforms the competitors that compute the extensions from scratch for single updates. In fact, on average, our technique is two orders of magnitude faster than them. Moreover, the harder the computation from scratch is, the larger the improvements are. That is, the results show that the improvements obtained for $\mathcal{S} \in \{\mathtt{st}, \mathtt{pr}\}$ go beyond those for $\mathcal{S} \in \{\mathtt{gr}, \mathtt{co}\}$.

– Our algorithm remains faster than the competitors even when recomputing an extension after performing a quite

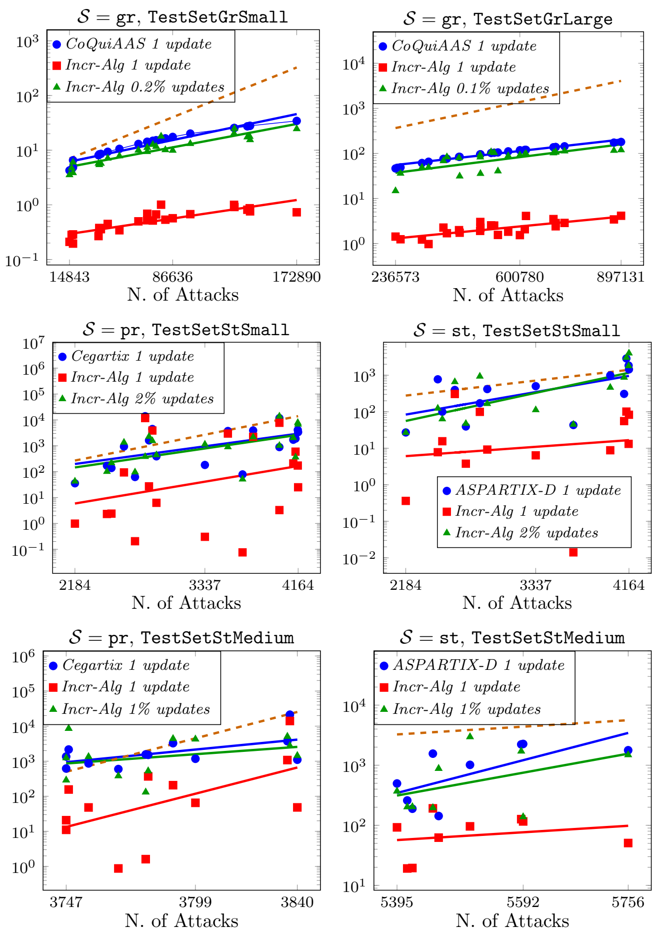

Figure 2: Run times (ms) of ICCMA solvers and *Incr-Alg* for different semantics $\mathcal{S}$ over different datasets (showed on the top of each graph) versus the number of attacks.

large number of updates simultaneously. In particular, in the graphs we show the threshold percentages of updated attacks up to which the incremental approach for multiple updates is faster than the computation from scratch.

– Finally, our experiments have also shown that for sets of updates regarding a relevant portion of the input AF (on average at least 1% of the attacks for $\mathcal{S} \in \{\texttt{st}, \texttt{pr}\}$ and 0,1% of the attacks for $\mathcal{S} \in \{\texttt{gr}, \texttt{co}\}$) recomputing extensions after applying them simultaneously is faster than recomputing extensions after applying them sequentially. Indeed, the green lines in the graphs are mostly below the (dashed) orange lines representing the run times of recomputing extensions after applying the updates sequentially.

6 Related Work

There have been several significant efforts coping with dynamics aspects of abstract argumentation. [Cayrol *et al.*, 2008; 2010] have addressed the problem of revising the set of extensions of an AF, and studied how the extensions can evolve when a new argument is considered. They focus on adding only one argument interacting with one initial argument (i.e. an argument which is not attacked by any other argument). [Bisquert *et al.*, 2013] have studied the evolution of the set of extensions after performing a change operation (addition/removal of arguments/interaction). Dynamic argumentation has been applied to decision-making of an autonomous agent by [Amgoud and Vesic, 2012], where it is studied how the acceptability of arguments evolves when a new argument is added to the decision system.

Other relevant works on dynamic aspects of AF include the following. [Baumann, 2011] have proposed an approach exploiting the concept of splitting of logic programs to deal with dynamic argumentation. The technique considers weak expansions of the initial AF, where added arguments never attack previous ones. [Baumann and Brewka, 2010] have investigated whether and how it is possible to modify a given AF so that a desired set of arguments becomes an extension, whereas [Oikarinen and Woltran, 2011] have studied equivalence between two AFs when further information (another AF) is added to both AFs. [Baumann, 2012] have focused on expansions where new arguments and attacks may be added but the attacks among the old arguments remain unchanged, while [Baumann, 2014] have characterized update and deletion equivalence, where adding/deleting arguments/attacks is allowed (deletions were not considered by [Oikarinen and Woltran, 2011; Baumann, 2012]).

7 Conclusion and Future Work

We introduced a technique enabling any non-incremental algorithm to be used as an incremental one for computing some extension of dynamic AFs. Our work advanced existing approaches for computing extensions of dynamic AFs from two standpoints: (i) we considered general forms of updates (i.e., not limited forms of updates such as for instance weak expansions [Baumann, 2011]) and (ii) we identified a tighter portion of the updated AF to be examined for recomputing the semantics. For instance, the reduced AF is significantly smaller than the *conditioned* AF (CAF) in [Liao *et al.*, 2011]. Additional experiments showed that the size of CAF is 91% of the size of the initial AF, while the size of the reduced AF is only 9%. Also, using the reduced AF is more efficient than using CAF: the run time of our technique turned out to be only 1% of the run time of using ICCMA solvers taking as input CAFs, that is, still two orders of magnitude faster.

Future work will be devoted to (i) applying our technique to other argumentation semantics and (ii) extending it to cope with other computational problems, such as enumerating all the extensions and deciding credulous/sceptical acceptance. As regard (i), our technique could be extended to deal with the ideal semantics [Dunne, 2009], for which experiments using ConArg [Bistarelli *et al.*, 2016] as non-incremental solver showed performance improvements. As for (ii), preliminary experiments showed that on average only about 4% of the extensions of the updated AF are lost if we start from the set of all extensions of the initial AF. However, as even the loss of one extension may change the justification status of arguments, we plan to investigate restrictions on the structure of AFs guaranteeing that no extension is lost.

References

[Amgoud and Vesic, 2012] Leila Amgoud and Srdjan Vesic. Revising option status in argument-based decision systems. *J. Log. Comput.*, 22(5):1019–1058, 2012.

[Baroni *et al.*, 2011] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.

[Baroni *et al.*, 2014a] Pietro Baroni, Guido Boella, Federico Cerutti, Massimiliano Giacomin, Leendert W. N. van der Torre, and Serena Villata. On the input/output behavior of argumentation frameworks. *AI*, 217:144–197, 2014.

[Baroni *et al.*, 2014b] Pietro Baroni, Massimiliano Giacomin, and Beishui Liao. On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method. *AI*, 212:104–115, 2014.

[Baumann and Brewka, 2010] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In *COMMA*, pages 75–86, 2010.

[Baumann, 2011] Ringo Baumann. Splitting an argumentation framework. In *LPNMR*, pages 40–53, 2011.

[Baumann, 2012] Ringo Baumann. Normal and strong expansion equivalence for argumentation frameworks. *AI*, 193:18–44, 2012.

[Baumann, 2014] Ringo Baumann. Context-free and context-sensitive kernels: Update and deletion equivalence in abstract argumentation. In *ECAI*, pages 63–68, 2014.

[Bench-Capon and Dunne, 2007] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *AI*, 171(1015):619 – 641, 2007.

[Bisquert *et al.*, 2013] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Characterizing change in abstract argumentation systems. In *Trends in Belief Revision and Argumentation Dynamics*, volume 48, pages 75–102. 2013.

[Bistarelli *et al.*, 2016] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. ConArg: A tool for classical and weighted argumentation. In *COMMA*, 2016.

[Boella *et al.*, 2009a] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in argumentation with single extensions: Abstraction principles and the grounded extension. In *ECSQARU*, pages 107–118, 2009.

[Boella *et al.*, 2009b] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in argumentation with single extensions: Attack refinement and the grounded extension. In *ArgMAS Workshop*, pages 150–159, 2009.

[Cayrol *et al.*, 2008] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Revision of an argumentation system. In *KR*, pages 124–134, 2008.

[Cayrol *et al.*, 2010] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in abstract argumentation frameworks: Adding an argument. *JAIR*, 38:49–84, 2010.

[Charwat *et al.*, 2015] Günther Charwat, Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - A survey. *AI*, 220:28–63, 2015.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AI*, 77(2):321–358, 1995.

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

[Dunne, 2009] Paul E. Dunne. The computational complexity of ideal semantics. *AI*, 173(18):1559–1591, 2009.

[Dvorák *et al.*, 2014] Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. *AI*, 206:53–78, 2014.

[Falappa *et al.*, 2011] Marcelo A. Falappa, Alejandro Javier Garcia, Gabriele Kern-Isberner, and Guillermo Ricardo Simari. On the evolving relation between belief revision and argumentation. *The Knowledge Engineering Review*, 26(1):35–43, 2011.

[Gaggl and Manthey, 2015] Sarah Alice Gaggl and Norbert Manthey. ASPARTIX-D ready for the competition, 2015.

[Greco and Parisi, 2016a] Sergio Greco and Francesco Parisi. Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks. In *ECAI*, pages 1668–1669, 2016.

[Greco and Parisi, 2016b] Sergio Greco and Francesco Parisi. Incremental computation of deterministic extensions for dynamic argumentation frameworks. In *JELIA*, pages 288–304, 2016.

[Kökciyan *et al.*, 2016] Nadin Kökciyan, Nefise Yaglikci, and Pinar Yolum. Argumentation for resolving privacy disputes in online social networks: (extended abstract). In *AAMAS*, pages 1361–1362, 2016.

[Lagniez *et al.*, 2015] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. CoQuiAAS: A constraint-based quick abstract argumentation solver. In *ICTAI*, pages 928–935, 2015.

[Liao *et al.*, 2011] Bei Shui Liao, Li Jin, and Robert C. Koons. Dynamics of argumentation systems: A division-based method. *AI*, 175(11):1790–1814, 2011.

[Modgil and Prakken, 2011] Sanjay Modgil and Henry Prakken. Revisiting preferences and argumentation. In *IJCAI*, pages 1021–1026, 2011.

[Oikarinen and Woltran, 2011] Emilia Oikarinen and Stefan Woltran. Characterizing strong equivalence for argumentation frameworks. *AI*, 175(14-15):1985–2009, 2011.

[Rahwan and Simari, 2009] Iyad Rahwan and Guillermo R. Simari. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009.