Extension Enforcement in Abstract Argumentation as an Optimization Problem

Sylvie Coste-Marquis **Sébastien Konieczny** **Jean-Guy Mailly** **Pierre Marquis**
CRIL, CNRS – Université d'Artois
Lens, France
{coste,konieczny,mailly,marquis}@cril.fr

Abstract

Change in abstract argumentation frameworks (AFs) is a very active topic. Especially, the problem of enforcing a set E of arguments, i.e., ensuring that E is an extension (or a subset of an extension) of a given AF F, has received a particular attention in the recent years. In this paper, we define a new family of enforcement operators, for which enforcement can be achieved by adding new arguments (and attacks) to F (as in previous approaches to enforcement), but also by questioning some attacks (and non-attacks) of F. This family includes previous enforcement operators, but also new ones for which the success of the enforcement operation is guaranteed. We show how the enforcement problem for the operators of the family can be modeled as a pseudo-Boolean optimization problem. Intensive experiments show that the method is practical and that it scales up well.

1 Introduction

Dung's seminal work on abstract argumentation [Dung, 1995] is the origin of a simple yet powerful setting to represent and reason about arguments. In this setting arguments are associated with the vertices of a directed graph, called an abstract argumentation framework (AF), and the corresponding arcs encode attacks. Several acceptability semantics have been defined in the objective of discriminating the arguments which can be jointly accepted.

AFs are useful for modeling and solving many problems, for instance to represent and reason about dialogs in a multi-agent system. Whatever the problem under consideration, the dynamics of AFs, i.e., how to make an AF evolve in light of new information, is an important issue. As such it has been addressed in many works in the recent years [Cayrol *et al.*, 2010; Bisquert *et al.*, 2011; Booth *et al.*, 2013; 2014; Coste-Marquis *et al.*, 2014a; Coste-Marquis *et al.*, 2014b].

In this paper the focus is laid on the enforcement problem: the key question is to determine whether it is possible to change an AF to ensure that a particular set of arguments is an extension, or at least is included in an extension. This problem has been studied in [Baumann and Brewka, 2010; Baumann, 2012]. In these works, there are some constraints on the allowed changes to the given AF: enforcing a set of arguments is achieved via the addition of some new arguments and some attacks between them and the arguments of AF; however no change amongst the attacks of AF is permitted.

Such enforcement operators are useful in many scenarios, especially when we consider an argumentative debate between several agents since the incoming of new arguments in the debate typically questions the existing extensions, and an agent can thus be interested in determining arguments to be added in order to enforce the set of arguments she likes. However, in many other scenarios, no new arguments are available for explaining the change, and one thus has to question the attacks between the arguments [Coste-Marquis *et al.*, 2014a; Coste-Marquis *et al.*, 2014b].

In this paper, we define a new family of enforcement operators, for which enforcement can be achieved by adding new arguments (and attacks) to the given AF F (as in previous approaches to enforcement), but also by questioning some attacks (and non-attacks) of F. This family includes previous enforcement operators as special cases, but also new ones for which the success of the enforcement operation is guaranteed. We show how the enforcement problem for the operators of this family can be modeled and achieved as a pseudo-Boolean optimization problem. Intensive experiments show that the method is practical and that it scales up well.

The paper is organized as follows. Some background on abstract argumentation is recalled in Section 2. Section 3 presents the definitions of the main families of enforcements defined in previous works from Baumann and Brewka, points out some of their limits in term of impossibility results and then defines a new family of enforcement operators, called argument-fixed enforcement, for which success is guaranteed. In Section 4, we explain how to achieve enforcement operations by solving optimization problems. Before the concluding section, Section 5 discusses the method used to implement enforcement operators, and gives experimental results.

2 Background

The following definitions come from [Dung, 1995].

Definition 1. *An abstract argumentation framework (AF) F is a directed graph $\langle A, R\rangle$ where A is a set of atomic entities called* arguments *and $R \subseteq A \times A$ is the* attack relation.

The intuitive meaning of the attack relation is that

$(a_i, a_j) \in R$ if when a_i is accepted by the agent, then a_j has to be rejected. A set of arguments $E \subseteq A$ is said to attack an argument a_i if and only if $\exists a_j \in E$ such that $(a_j, a_i) \in R$. An argument a_i (respectively a set of arguments E) defends the argument a_j against a_k such that $(a_k, a_j) \in R$ if a_i (respectively E) attacks a_k.

In order to characterize the arguments to be accepted, Dung pointed out several *acceptability semantics*, which aim at defining *extensions*: an extension is a set of arguments which can be jointly accepted by the agent. Whatever the semantics σ, $Ext_\sigma(F)$ denotes the set of σ-extensions of the AF F. The various semantics reflect some properties which ought to be satisfied by the extensions. For instance, $E \subseteq A$ is a conflict-free set in $F = \langle A, R\rangle$ if and only if there is no $a_i, a_j \in E$ such that $(a_i, a_j) \in R$. Then, $E \subseteq A$ is a complete extension of $F = \langle A, R\rangle$ if and only if E is conflict-free and E containts each $a_k \in A$ which is defended by E. The grounded extension is the minimal (with respect to $\subseteq$) complete extension. $E \subseteq A$ is a stable extension of $F = \langle A, R\rangle$ if and only if E is conflict-free and E attacks every argument $a_k \in A \backslash E$.

3 Extension Enforcement

Enforcing a set of arguments E is defined in [Baumann and Brewka, 2010] as a change from an AF F to another one F' such that E is an extension of F' or is included in an extension of F'. Several enforcement methods are presented, based on the notion of *expansion* of an AF. An expansion is the addition of new arguments and new attacks to an AF, respecting some constraints. The enforcement of E in F is then defined as an expansion of F such that E is an extension of it. Three kinds of expansion are considered:

- *Normal* expansion: some arguments are added, with some new attacks such that at least one of the new arguments is concerned by each new attack (there is no change in the attacks between former arguments).
- *Weak* expansion is a normal expansion such that no new attack is directed from a new argument to a former one.
- *Strong* expansion is a normal expansion such that no new attack is directed from a former argument to a new one.

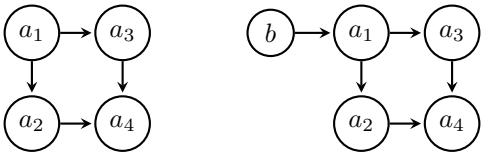
Beyond the nature of the expansion, two additional parameters must be made precise in order to define enforcement operators. First, enforcement can be *strict* when the expected set of arguments has to be exactly an extension of the output AF or *non-strict* when the set of arguments has to be included in an extension of the output AF. Then, enforcement can be *conservative* when the semantics stays the same one or *liberal* when the semantics may change. In the following, we define enforcement operators which can be used for both conservative and liberal enforcement, since the semantics associated with the input AF is not specified in the definition of the operators. For the sake of brevity, we focus only on the conservative enforcement situation.

Definition 2. *Let $F = \langle A, R\rangle$ be an AF, σ an acceptability semantics, and $E \subseteq A$ a set of arguments. The* normal (respectively normal strict) enforcement operator $+^N_\sigma$ *(resp. $+^N_{\sigma,s}$) is defined as a mapping from F and E to an AF $F' = \langle A \cup A', R \cup R_{A'}\rangle$, with $R_{A'}$ a set of attacks (a_i, a_j) such that $a_i \in A'$ or $a_j \in A'$, and such that E is included in (resp. is exactly) an extension of F'. Moreover,*

- *if $R_{A'} \cap (A' \times A) = \emptyset$, then $+^{N,W}_\sigma$ (resp. $+^{N,W}_{\sigma,s}$) is a* weak (resp. strict weak) enforcement operator*;*
- *if $R_{A'} \cap (A \times A') = \emptyset$, then $+^{N,S}_\sigma$ (resp. $+^{N,S}_{\sigma,s}$) is a* strong (resp. strict strong) enforcement operator.

Let us illustrate the strong enforcement approach.

Example 1. *Let F be the AF given in Figure 1(a). Its set of stable extensions is $Ext_{st}(F) = \{\{a_1, a_4\}\}$. The expected extension to be enforced is $E = \{a_2, a_3\}$. A possible strong enforcement is presented in Figure 1(b): the stable extensions of F_1 are $Ext_{st}(F_1) = \{\{a_2, a_3, b\}\}$, which contains E.*



(a) Input AF F (b) Enforced AF F_1

Figure 1: Strong enforcement process

Importantly, whatever the normal enforcement operator under consideration, it must be noted that enforcement may fail. As a simple example, let us consider $E = \{a_1, a_2\}$ in an AF $F = \langle A, R\rangle$ such that $(a_1, a_2) \in R$. It is obviously impossible to enforce E with any of the enforcement operators described previously. Theorem 2 and Theorem 3 from [Baumann and Brewka, 2010] give some more elaborated impossibility results about strict enforcement. An interesting result from [Baumann and Brewka, 2010] states that for each AF F, it is possible to enforce any set of arguments E which is conflict-free in F with a non-strict strong enforcement, and also guarantees that adding a single new argument is enough. This means that a non-strict strong enforcement can be performed with any singleton A'. This will be useful to define logical encodings suited to enforcement (see Section 4).

Note that the presence of conflicts in the set E of arguments to be enforced is a sufficient, yet unnecessary condition for normal enforcement to fail. In order to make it more formal, let us first introduce the notion of non-trivial set of arguments with respect to a given semantics:

Definition 3. *Let $F = \langle A, R\rangle$ be an AF, and σ a semantics. $E \subseteq A$ is a σ non-trivial set of arguments in F if and only if E is conflict-free in F and $E \notin Ext_\sigma(F)$.*

Assuming the set E of arguments to be enforced to be σ non-trivial is a way to avoid the trivial cases when enforcement is already satisfied because E is a σ-extension of F or impossible because of conflicts. However, it does not prove sufficient for preventing from failure for every semantics:

Proposition 1. *For every $F = \langle A, R\rangle$ and $E \subseteq A$ a stable non-trivial set in F, there is no strict enforcement of E in F with respect to the stable semantics.*

Proposition 2. *For every $F = \langle A, R\rangle$, and $E \subseteq A$ a complete non-trivial set in F,*

1. *if E does not defend itself against each attacker, then there is no strict enforcement of E in F with respect to the complete semantics.*
2. *else, if E defends some argument $a_i \in A \backslash E$, then*
   (a) *there is no strict weak enforcement of E in F with respect to the complete semantics.*
   (b) *if odd-length cycles are not allowed, then there is no strict strong enforcement of E in F with respect to the complete semantics.*

Proposition 3. *For every $F = \langle A, R\rangle$ and $E \subseteq A$ a grounded non-trivial set in F, if $Ext_{gr}(F) = \{\emptyset\}$, then there is no strict enforcement of E in F with respect to the grounded semantics.*

Argument-Fixed Enforcement. In the previous approaches for enforcing a set of arguments, it is supposed that new arguments can be added, and that interactions between the existing arguments do not change. This method is particularly sensible when enforcement is supposed to be the result of a dialog: given an AF representing the state of a dialog, an agent adds new arguments if she wants to convince the other agent to accept a given set of arguments as an extension. Forbidding any change over the initial attacks of the framework is the reason of the above impossibility results. Interestingly, the converse case, i.e., considering situations where the set of arguments cannot change, but the attack relation is subject to evolutions, also makes sense. It is sensible, for instance, when a set of arguments is observed to be an extension in the output of an argumentation process, but does not correspond to the output of the own AF of an agent. In such a case, without the knowledge of some new arguments, the agent has to change her beliefs about the attack relation to be consistent with the observed set of arguments.

Definition 4. *Let $F = \langle A, R\rangle$ be an AF, σ an acceptability semantics, and $E \subseteq A$ a set of arguments. The* argument-fixed (resp. strict argument-fixed) enforcement operator $+^A_\sigma$ *(resp. $+^A_{\sigma,s}$) is defined as a mapping from F and E to an AF $F' = \langle A, R'\rangle$, with $R' \subseteq A \times A$, and such that E is included in (resp. is exactly) an extension of F'.*

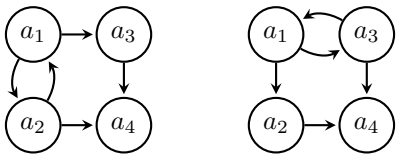
The argument-fixed operators guarantee the success of enforcement, even in the strict case:

Proposition 4. *Let $F = \langle A, R\rangle$ be an AF, σ and acceptability semantics and $E \subseteq A$ a set of arguments. There is a a strict enforcement F' of E in F.*

Of course, both ideas (adding arguments, and changing the attacks) can be combined:

Definition 5. *Let $F = \langle A, R\rangle$ be an AF, σ an acceptability semantics, and $E \subseteq A$ a set of arguments. The* general (resp. strict general) enforcement operator $+_\sigma$ *(resp. $+_{\sigma,s}$) is defined as a mapping from F and E to an AF $F' = \langle A \cup A', R'\rangle$, with $R' \subseteq A \times A'$, and such that E is included in (resp. is exactly) an extension of F'.*

Example 2. *Let us consider again the AF F described in Figure 1(a). We gave an example of non-strict strong enforcement, but as shown by Proposition 1, it is impossible to perform a strict enforcement under the stable semantics using Baumann's approaches (normal, strong and weak). Let us use the argument-fixed enforcement operator to obtain a strict enforcement of the set of argument $E = \{\{a_2, a_3\}\}$. A possible result is the AF F_2 described in Figure 2(a), whose stable extensions are $Ext_{st}(F_2) = \{\{a_1, a_4\}, \{a_2, a_3\}\}$, and so E is enforced as a stable extension of the result. Another one is F_3 given in Figure 2(b), whose set of stable extensions is the same one: $Ext_{st}(F_3) = \{\{a_1, a_4\}, \{a_2, a_3\}\}$.*



(a) The AF F_2 (b) The AF F_3

Figure 2: Two possible results of the argument-fixed enforcement

Minimal Change. As explained in Proposition 4, the possibility to enforce a set of argument is ensured when changes on the attack relation are allowed. From a practical point of view, it offers a success guarantee, which is a valuable property for an enforcement operator. Another expected property is minimal change, borrowed from belief revision. Enforcement processes can lead to several results, and the enforcement operators defined previously have to select one of the possible AFs as the result. Considering minimal change during the enforcement means that the chosen AF has to be as close as possible to the initial AF. [Baumann, 2012] already studies such a notion of closeness for the normal enforcement approaches. He defines minimal change as minimization of the number of attacks which are added to the AF during the enforcement process. We generalize this notion of minimal change, using the well-known Hamming distance to measure how much two AFs are different.

Definition 6. *The Hamming distance d_h between two AFs $F = \langle A, R\rangle$ and $F' = \langle A', R'\rangle$ is defined by:*

$$d_h(F, F') = |(R \backslash R') \cup (R' \backslash R)|$$

For every enforcement operator $+$, the minimal change version $\underset{\min}{+}$ is such that the selected output AF F' minimizes the Hamming distance from the input AF F amongst the set of AFs which are possible outputs of $+$.

4 Enforcement as Satisfaction and Optimization

A first observation is that enforcing a set of arguments while limiting the number of allowed changes in the attack relation is computationally demanding in the general case:

Proposition 5. *Let $F = \langle A, R\rangle$ be an AF, $E \subseteq A$, and an integer k. Determining whether it is possible to enforce E in F under the stable semantics with at most k changes (addition or removal) of attacks is* NP*-hard.*

Proposition 5 ensures that (unless $\mathsf{P} = \mathsf{NP}$) there is no polynomial-time algorithm to perform minimal change enforcement in the general case. For this reason, it makes sense to tackle the enforcement (resp. minimal change enforcement) issue using algorithms developed for solving (resp. optimizing) NP-hard problems. This is what we do in the following: we reduce enforcement to a propositional satisfiability problem, and minimal change enforcement to a pseudo-Boolean optimization problem.

Enforcement as Boolean Satisfaction. Our translation-based approach is based on the possibility to associate an AF F and a semantics σ with a propositional formula such that the models of the formula correspond exactly to σ-extensions of F.

Definition 7. *Given F an AF and σ a semantics, Φ_σ^F is a propositional formula built upon the set of Boolean variables $\{x_a \mid a \in A\}$, such that $\{x_{a_1}, \ldots, x_{a_k}\}$ is a model of Φ_σ^F if and only if $\{a_1, \ldots, a_k\}$ is a σ-extension of F.*

In the following, for a matter of simplification and since no ambiguity is possible, we write the formulae using a_i symbols instead of x_{a_i}. We focus on the encoding Φ_{st} of the stable extension, as given in [Besnard and Doutre, 2004]. Given $F = \langle A, R\rangle$, Φ_{st} is defined by

$$\Phi_{st} = \bigwedge_{a_i \in A} [a_i \Leftrightarrow (\bigwedge_{a_j:(a_j,a_i)\in R} \neg a_j)]$$

Then, checking if a set of arguments E is a stable extension of F is equivalent to checking the satisfiability of the formula $\Phi^E_{st,s} = \Phi_{st} \wedge (\bigwedge_{a_k \in E} a_k) \wedge (\bigwedge_{a_l \notin E} \neg a_l)$. To perform non-strict enforcement, a way to determine whether E is included in an extension is required. Dropping the conjunct $(\bigwedge_{a_l \notin E} \neg a_l)$ from the formula $\Phi^E_{st,s}$ gives precisely the formula Φ^E_{st} we need.

In order to link the semantics with the structure of the graph in the models of the formula, we introduce Boolean variables att_{a_i,a_j} meaning that there is an attack from a_i to a_j in F. The previous formulae are generalized into:

$$\begin{aligned}\Phi^{A,E}_{st} = \; & \textstyle\bigwedge_{a_i \in A}[a_i \Leftrightarrow (\bigwedge_{a_j \in A}(att_{a_j,a_i} \Rightarrow \neg a_j)] \\ & \wedge(\textstyle\bigwedge_{a_k \in E} a_k)\end{aligned}$$

and

$$\begin{aligned}\Phi^{A,E}_{st,s} = \; & \textstyle\bigwedge_{a_i \in A}[a_i \Leftrightarrow (\bigwedge_{a_j \in A}(att_{a_j,a_i} \Rightarrow \neg a_j)] \\ & \wedge(\textstyle\bigwedge_{a_k \in E} a_k) \wedge (\bigwedge_{a_l \notin E} \neg a_l)\end{aligned}$$

Clearly, propagating the truth values of the variables att_{a_i,a_j} in those formulae is enough to recover the previous formula $\Phi^E_{st,s}$ and the non-strict counterpart. This formula is the basis of our propositional encoding of extension enforcement operators. It remains to introduce two functions allowing to "decode" such a formula and get AFs:

- $Proj^A_{att}(\Phi) = \{m \cap \{att_{a_i,a_j} \mid a_i, a_j \in A\} \mid m \models \Phi\}$ is the sets of models of the formula Φ projected onto the att_{a_i,a_j} variables.
- $arg^A(m) = \langle A, R\rangle$ such that $(a_i, a_j) \in R$ if and only if $att_{a_i,a_j} \in m$, with m a model projected onto the att_{a_i,a_j} variables, is the AF corresponding to the assignment of the att_{a_i,a_j} variables. Then, with M a set of such models, $arg^A(M) = \{arg^A(m) \mid m \in M\}$.

We also need an encoding for the structure of an AF $F = \langle A, R\rangle$:

$$struct_{A'}(F) = (\bigwedge_{(a_i,a_j)\in R} att_{a_i,a_j}) \wedge (\bigwedge_{(a_i,a_j)\notin R} \neg att_{a_i,a_j})$$

where $a_i, a_j \in A \cup A'$. $struct(F)$ is a notation for $struct_\emptyset(F)$.

Finally, $\delta : \{F_1, \ldots, F_k\} \to F_j$ such that $F_j \in \{F_1, \ldots, F_k\}$ is a tie-break rule which selects a single AF from a set of AFs.

Now, every enforcement operator defined in the previous section can be encoded as a satisfaction problem on a propositional formula. Indeed, by construction, every model of the formula $\Phi_\sigma^{A\cup A',E}$, when projected onto the att_{a_i,a_j} variables, gives an AF which is a normal enforcement of E. We only need to state the right constraints for ensuring that strong (resp. weak) enforcement operators are reached. In order to avoid the introduction of new arguments and get argument-fixed operators, considering the formula $\Phi_\sigma^{A,E}$ as the encoding proves enough. Similarly, the formulae $\Phi_{\sigma,s}^{A\cup A',E}$ and $\Phi_{\sigma,s}^{A,E}$ can be used to define the strict counterparts of the enforcement operators.

Definition 8. *For any AF $F = \langle A, R\rangle$, any set of arguments $E \subseteq A$, any semantics σ, and $X = \sigma$ or $X = \sigma, s$:*

$F +^N_X E = \delta(arg^{A\cup A'}(Proj^{A\cup A'}_{att}(\Phi^{A\cup A',E}_X \wedge struct(F)))$

$F +^{N,W}_X E = \delta(arg^{A\cup A'}(Proj^{A\cup A'}_{att}(\Phi^{A\cup A',E}_X \wedge struct(F) \wedge(\bigwedge_{(a_i,a_j)\in A'\times A} \neg att_{a_i,a_j}))))$

$F +^{N,S}_X E = \delta(arg^{A\cup A'}(Proj^{A\cup A'}_{att}(\Phi^{A\cup A',E}_X \wedge struct(F) \wedge(\bigwedge_{(a_i,a_j)\in A\times A'} \neg att_{a_i,a_j}))))$

$F +^A_X E = \delta(arg^A(Proj^A_{att}(\Phi^{A,E}_X)))$

$F +_X E = \delta(arg^{A\cup A'}(Proj^{A\cup A'}_{att}(\Phi^{A\cup A',E}_X))).$

For any of these enforcement operators $+$, any AF F and any set of arguments E, $Enc(F + E)$ denotes the corresponding propositional encoding. For instance, $Enc(F +^N_\sigma E)$ is the propositional formula $\Phi_\sigma^{A\cup A',E} \wedge struct(F)$. Using any SAT solver to find a model of $Enc(F + E)$ and then decoding the truth values of the att_{a_i,a_j} variables is a way to determine an enforcement of E in F.

Minimal Change Enforcement as Pseudo-Boolean Optimization. As explained previously, [Baumann, 2012] considers a notion of minimal change enforcement. In his work, minimality refers to the minimality of the number of attacks to be added to the AF when performing the normal expansion. A possible way to ensure minimal change is to define a particular tie-break rule δ for selecting one of the resulting AFs which is minimal. In order to take advantage of some available optimization software, an alternative approach is to

encode the minimality criterion via a pseudo-Boolean objective function:

$$newAtt(A \cup A') = \sum_{(a_i,a_j)\in((A\cup A')\times(A\cup A'))\setminus(A\times A)} att_{a_i,a_j}$$

Of course, for strong and weak enforcement operators, this representation of the objective function can be simplified since the att_{a_i,a_j} variables corresponding to the forbidden attacks are known to be *false*.

Minimal change for argument-fixed and general enforcement is not easy to be encoded directly using the available Boolean variables. In order to get the expected encoding, we consider additional variables representing the state of the AF before the enforcement; one then minimizes the number of differences between the truth values of these variables and the corresponding ones in the new AF. Formally, for every pair of arguments $(a_i, a_j) \in (A \cup A') \times (A \cup A')$, the Boolean variable $prev_{a_i,a_j}$ is *true* if and only if $(a_i, a_j) \in R$. So, $prev_{a_i,a_j} \oplus att_{a_i,a_j}$, where $\oplus$ is the usual exclusive-or connective, gives the information about the change on the attack (a_i, a_j): if there was previously an attack from a_i to a_j, and this attack is no longer present after the enforcement, $prev_{a_i,a_j} \oplus att_{a_i,a_j}$ is *true*. It is also *true* if there was no attack before the enforcement, and there is one after the enforcement. The encoding of the structure of the AF F must thus be updated to take account for the $prev_{a_i,a_j}$ variables:

$$struct^{prev}_{A'} = (struct_{A'}(F)_{|att_{a_i,a_j}\leftarrow prev_{a_i,a_j}})$$

Once this is done, minimizing the differences on the attack relation is equivalent to minimizing the objective function

$$attChange(A\cup A') = \sum_{a\in(A\cup A'),b\in(A\cup A')} prev_{a_i,a_j} \oplus att_{a_i,a_j}$$

Clearly, this sum counts 1 for every attack (a_i, a_j) in the output AF concerning an argument of A', because $prev_{a_i,a_j}$ is always *false* if $a_i \in A'$ or $a_j \in A'$. So the approach can be used in the case of general enforcement.

We now sum up the definitions of the minimal change versions of the enforcement operators:

Definition 9. *For any AF $F = \langle A, R\rangle$, any set of arguments $E \subseteq A$,*

- *if $+$ is any enforcement operator among the normal, strong and weak enforcement operators (and their strict counterparts), then enforcing the set of arguments E in F is equivalent to satisfying $Enc(F + E)$ while minimizing $newAtt(A \cup A')$;*
- *if $+$ is any enforcement operator among the argument-fixed and general enforcement operators (and their strict counterparts), then enforcing the set of arguments E in F is equivalent to satisfying $Enc(F + E) \wedge struct^{prev}_{A'}(F)$ while minimizing $attChange(A \cup A')$.*

We notice that using the second optimization problem would prove enough for each enforcement operator. But since this approach requires the addition of Boolean variables $prev_{a_i,a_j}$, we do not use it when it is not mandatory, to avoid any loss of computational efficiency.

The formal setting suited to our optimization problem is pseudo-Boolean (PB) optimization, which is an extension of Boolean satisfiability.

Definition 10. *Given a set of Boolean variables $V = \{x_1, \ldots, x_n\}$ and a mapping $\mathcal{O} : \{0, 1\}^n \mapsto \mathbb{R}$, a* PB-*Opt problem $\mathcal{P} = (\mathcal{C} = \{c_1, \ldots, c_m\}, \mathcal{O})$ on V is the search for an assignment of every variable in V such that the contraints*

$$\begin{array}{rl} c_1: & w^1_1 x_1 + \cdots + w^1_n x_n \geq k^1 \\ \vdots & \vdots \\ c_m: & w^m_1 x_1 + \cdots + w^m_n x_n \geq k^m \end{array}$$

are satisfied and the objective function $\mathcal{O}$ reaches its optimal value.

In our case, the optimal value of the objective function is its minimal value. It is well-known that any propositional formula can be turned into an equivalent conjunctive normal form formula (CNF), and any clause of a CNF formula can be rewritten as a PB constraint: the clause $x_1 \vee x_2 \vee \cdots \vee x_n$ is satisfied if and only if the PB constraint $x_1 + x_2 + \cdots + x_n \geq 1$ is satisfied. Thus the optimization problem described previously can be rewritten easily in the PB setting.

5 Experimental Results

In our experimental study, we focused on the minimal change enforcement problem. We implemented the family of enforcement operators described in this paper, using the well-known tool `CPlex` [IBM, 2014] as the underlying optimization engine. For space reasons, we present only the obtained results for three approaches: the non-strict strong operator from [Baumann and Brewka, 2010], and both the strict and non-strict versions of our argument-fixed enforcement operator. In each case, the semantics used is the stable one.

The empirical protocol we considered is as follows. We focused on some random AFs [Dvorák *et al.*, 2011; 2014]. Given a set of n arguments, each attack between two arguments is generated using a fixed probability p. In our experiments n varies up to 500 arguments. For each n, the graphs are divided into four families, corresponding to four values of p. We used families of AFs from [Dvorák *et al.*, 2011], where $p \in \{0.4, 0.65, 0.9\}$. We also generated AFs with a probability $p = 0.1$. It appears in the experiments that this choice of p does not change significantly the performances of the translation-based enforcement algorithm, so the reported results are for $p = 0.1$ only.

We have computed the minimal change enforcement of sets E of arguments in AFs F containing n arguments with $n \in \{200, 300, 400, 500\}$. For each AF F with n arguments, we considered sets E of arguments to be enforced of cardinality m, m varying between 1 and $\frac{35}{100}n$. For each pair of values (n, m), we generated 10 enforcement requests.[1] On Figure 3, the y-coordinate of each point of the curves corresponds to the average computation time over all the pairs (F, E) which have been considered, where the number n of arguments of F is reported on the x-axis.

[1] We call "enforcement request" the set E of arguments expected to be an extension.

The first interesting result stemming from our experimentations is that enforcement looks feasible in practice on such randomly generated AFs, which was not obvious given that enforcement is NP-hard. As illustrated by Figure 3, the computation time increases reasonably with the number n of arguments, up to a mean value of 13.76 seconds (std = 0.48) obtained for AFs with 500 arguments, when strict argument-fixed enforcement is considered ($+$-curve), and up to a mean of 16.61 seconds (std = 5.31) when strong enforcement is considered ($\times$-curve).

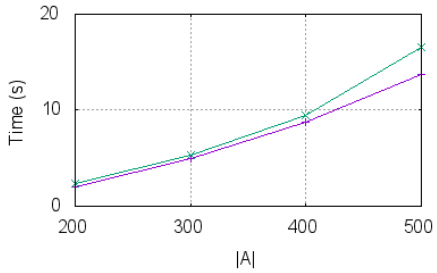
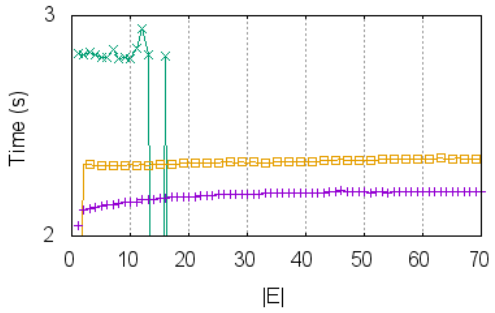


Figure 3: Average time for strong ($\times$-curve) and strict argument-fixed ($+$-curve), n varying from 200 to 500

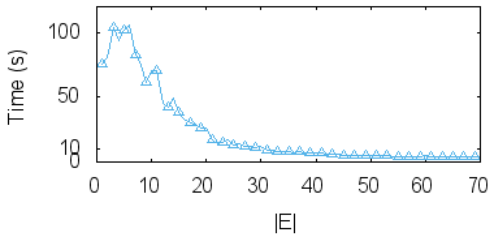
Then, we compared the three different approaches on families of AFs with 200 arguments, letting the cardinality m of E to vary from 1 to 70. The aim of this comparison is to study the impact of the cardinality of E on the enforcement operators behaviors. We did not discard trivial sets from the experiments, since our approaches can delete attacks, rendering a conflicting set conflict-free. This allows us to illustrate the failure rate of strong enforcement, which is unsurprisingly high, since enforcement is impossible as soon as the enforcement request is not conflict-free in the input AF. Indeed, with a probability p for an attack to occur between two arguments, the probability for a set of arguments E of cardinality m to be conflict-free is $(1-p)^m$. So, the greater the cardinality of the enforcement request, the lower the probability for enforcement to be possible. In particular, in our experiments strong enforcement always fails when $m > 20$; clearly, the failure rate of strong enforcement grows exponentially with m.

We compared the enforcement computing times for the three approaches (see Figure 4). The $\times$-curve represents the average time to realize the strong enforcement, while the $\square$-curve corresponds to the time needed by the algorithm to report the failure when enforcement is impossible. For strong and strict argument-fixed ($+$-curve) enforcement, it appears that the time needed for computing the result is almost always the same whatever the cardinality of the enforcement request and the probability of attacks in the graph: between 2 and 3 seconds. The cardinality has more influence on the non strict argument-fixed enforcement operator, the smallest enforcement requests being harder to compute. When the cardinality grows, the computing time decreases to a few seconds.

Lastly, we have measured the effort required in term of change (i.e., the number of attacks to be added or deleted) for enforcing E in the AF (see Figure 5). Clearly, the effort



(a) Strict argument-fixed ($+$-curve) enforcement, and success ($\times$-curve) and failure ($\square$-curve) for strong enforcement



(b) Non strict argument-fixed enforcement

Figure 4: Average time, $n = 200$, m varying from 1 to 70

needed grows up with the cardinality of the enforcement request for strong ($\times$-curve) and non strict argument-fixed enforcement ($\triangle$-curve). The curve for strong enforcement stops much before the other ones (at 14 arguments) because of the high failure rate we mentioned. Until this point, as one can observe, the $\times$-curve is almost identical to the $\triangle$-curve. Strict argument-fixed enforcement ($+$-curve) requires much more change on the attack relation. As the enforced set is expected to be exactly a stable extension, even a small set of arguments requires the addition of many attacks to be enforced. For instance, with $E = \{a_i\}$, it is required that a_i attacks each other argument to ensure that E is a stable extension. When the cardinality m of E grows up, the efforts required by the strict and by the non strict versions of the argument-fixed operator come closer.

6 Conclusion

In this work, we have investigated the problem of enforcing a set of arguments as an extension of an AF. Our contribution is manyfold. First, we have shown that existing approaches to enforcement may fail, even when the set of arguments to be enforced is conflict-free. To overcome this weakness, new enforcement methods for which the success of the process can be guaranteed have been defined. For each of these methods, we designed some Boolean encodings which allow to take advantage of satisfaction and optimization solvers for the enforcement purpose. We used a well-known optimization tool to implement a library of enforcement operators, and we experimented some of them on a large class of benchmarks. The experimentations showed the approach to be practical and to

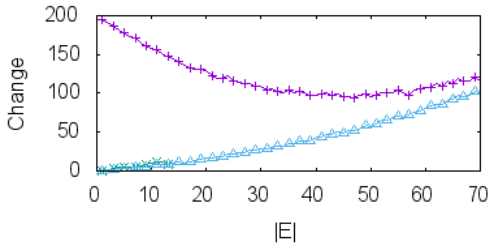


Figure 5: Average change for strong ($\times$-curve), strict argument-fixed ($+$-curve) and non-strict argument-fixed ($\triangle$-curve) enforcement

scale up well.

It is worth noticing that some other kinds of change operations are specific extension enforcements. For instance, credulous explanation in [Booth *et al.*, 2014] is the enforcement of a singleton. Similarly, some goal-oriented changes from [Kontarinis *et al.*, 2013] and some revision operators from [Coste-Marquis *et al.*, 2014a; Coste-Marquis *et al.*, 2014b] are enforcement operators. Our enforcement approach thus proves also useful for achieving such kinds of changes in AFs.

This work opens several perspectives for further research. As far as we know, none of the existing works about change in argumentation frameworks has led to the implementation of some (quite efficient) piece of software. However, implementing practical argumentation systems is currently a hot topic for the community (in the same vein, see the organization of a competition of argumentation solvers [Thimm and Villata, 2015]). Indeed, the design of our enforcement software comes from the same will to make available argumentation reasoning tools, which is nowadays a necessary step to push forward the domain. So, we plan to encode and implement enforcement operators for other semantics. Some further extensions of the setting will also be envisioned, like minimal change on arguments statuses or the addition of integrity constraints.

Acknowledgments

This work benefited from the support of the project AMANDE ANR-13-BS02-0004 of the French National Research Agency (ANR).

References

[Baumann and Brewka, 2010] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In *COMMA'10*, pages 75–86, 2010.

[Baumann, 2012] Ringo Baumann. What does it take to enforce an argument? Minimal change in abstract argumentation. In *ECAI'12*, pages 127–132, 2012.

[Besnard and Doutre, 2004] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In *NMR'04*, pages 59–64, 2004.

[Bisquert *et al.*, 2011] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint Cyr Bannay, and Marie-Christine Lagasquie-Schiex. Change in argumentation systems: exploring the interest of removing an argument. In *SUM'11*, pages 275–288, 2011.

[Booth *et al.*, 2013] Richard Booth, Souhila Kaci, Tjitze Rienstra, and Leendert van der Torre. A logical theory about dynamics in abstract argumentation. In *SUM'13*, pages 148–161, 2013.

[Booth *et al.*, 2014] Richard Booth, Dov M. Gabbay, Souhila Kaci, Tjitze Rienstra, and Leendert W. N. van der Torre. Abduction and dialogical proof in argumentation and logic programming. In *ECAI'14*, pages 117–122, 2014.

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

[Coste-Marquis *et al.*, 2014a] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. On the revision of argumentation systems: Minimal change of arguments statuses. In *KR'14*, pages 52–61, 2014.

[Coste-Marquis *et al.*, 2014b] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. A translation-based approach for revision of argumentation frameworks. In *JELIA'14*, pages 397–411, 2014.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[Dvorák *et al.*, 2011] Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran, 2011. see `http://www.dbai.tuwien.ac.at/research/project/argumentation/dynpartix/examples/`.

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

[IBM, 2014] IBM. IBM ILOG CPLEX Optimization Studio: Optimization model development toolkit for mathematical and constraint programming, 2014. See `http://www.ibm.com/software/products/en/ibmilogcpleoptistud/`.

[Kontarinis *et al.*, 2013] Dionysios Kontarinis, Elise Bonzon, Nicolas Maudet, Alan Perotti, Leon van der Torre, and Serena Villata. Rewriting rules for the computation of goal-oriented changes in an argumentation system. In *CLIMA XIV*, pages 51–68, 2013.

[Thimm and Villata, 2015] Matthias Thimm and Serena Villata. First International Competition on Computational Models of Argumentation (ICCMA'15), 2015. see `http://argumentationcompetition.org/2015/`.