Computational Models of Argument
H. Prakken et al. (Eds.)

© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200503

Computing Strongly Admissible Sets

Wolfgang DVOŘÁK and Johannes P. WALLNER
TU Wien, Vienna, Austria

Abstract. In this work we revisit computational aspects of strongly admissible semantics in Dung's abstract argumentation frameworks. First, we complement the existing complexity analysis by focusing on the problem of computing strongly admissible sets of minimum size that contain a given argument and providing NP-hardness as well as hardness of approximation results. Based on these results, we then investigate two approaches to compute (minimum-sized) strongly admissible sets based on Answer Set Programming (ASP) and Integer Linear Programming (ILP), and provide an experimental comparison of their performance.

Keywords. abstract argumentation, strongly admissible, computational complexity, answer set programming, integer linear programming

1. Introduction

A key part to argumentative reasoning in Artificial Intelligence (AI) [3,6] are argumentation frameworks (AFs) due to Phan Minh Dung [14], which provide a formal approach to represent arguments as abstract entities together with directed conflicts (attacks) among the arguments. Semantics of AFs define criteria which sets of arguments can be deemed acceptable, where the notions of conflict-freeness and defense of arguments prove to be essential. A set of arguments is conflict-free if no arguments in the set are conflicting, and defense requires that each attack onto a set is counter-attacked from inside the set.

A particularly cautious representative of AF semantics is the grounded semantics that includes all unattacked (undoubted) arguments and each argument that can be iteratively defended from these unattacked arguments in the grounded extension. Almost all major semantics of AFs contain the arguments of the grounded extension [14]. An important reasoning task for the grounded semantics is to verify whether a queried argument is part of the grounded extension, or not. Notably, to answer this question, not all arguments within the grounded extension are necessary.

Example 1. *Consider an AF as shown in Figure 1. Say we desire to understand the acceptability of argument a under grounded semantics. The unique grounded extension of this AF is* $\{a, c, g, h, j, l\}$ *which answers this question positively. Yet, not all arguments are required to answer this question: e.g., argument c is sufficient to counter argument b and to defend a, and g can be used to defend argument c from its three attackers.*

A general observation from the preceding example can be made, and was formalized in the literature: one can define dialectical proof procedures, or game-theoretic notions, that specify which parts of the grounded extension suffice to witness containment in the grounded extension. Under a game-theoretic perspective a proponent only needs to con-

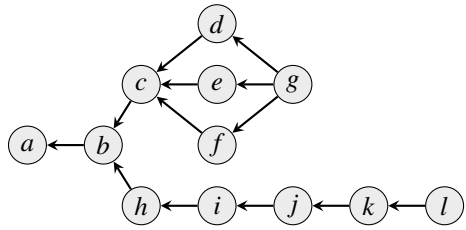

Figure 1. Two strongly admissible sets containing a: $\{a,c,g\}$ and $\{a,h,j,l\}$

sider arguments g and c to defend a against each possible counter-argument. Such game-theoretic notions for the grounded semantics were studied and resulted in the Standard Grounded Game [25,23], and the Grounded Discussion Game [8].

Importantly, so-called strongly admissible sets [4,9] turned out to be key components for winning strategies for such games. Admissible sets are conflict-free sets of arguments where each argument in the set is defended by the set. Strongly admissible sets, in contrast, require, intuitively speaking, that defense is "rooted" in unattacked arguments. In addition, strongly admissible sets were not only shown to be viable for explaining acceptance under grounded semantics, but, recently, also shown to be useful for explaining certain notions of non-acceptance [26].

Interestingly, while several papers provide results for strongly admissible sets [4, 9,5,15], in the literature strongly admissible sets were not yet studied in-depth from a computational perspective. While the grounded extension, which can be computed in polynomial time [16], would suffice to give a (maximal) strongly admissible set, explanations, such as via winning strategies for games, benefit from only requiring as few arguments as possible. Surprisingly, while all common reasoning tasks for the grounded semantics are polynomial time decidable, we show that finding a strongly admissible set of minimum size containing a queried argument is, in fact, a complex problem: we show that a natural decision variant is NP-complete. Even more, we show that approximating minimum-sized strongly admissible sets containing a queried argument remains NP-hard.

Our main contributions in this paper are as follows.

- We show NP-completeness of deciding whether there is a strongly admissible set of size at most a given integer that contains a queried argument. Moreover, we also turn some *in* P results of [9] to P-completeness results.
- We tighten the complexity landscape by showing NP-hardness for approximating strongly admissible sets of minimum size.
- We provide two computational approaches inspired by the success of the "reduction approach" to AF reasoning [12]: (a) an encoding in Answer Set Programming (ASP) and (b) an Inter Linear Programming (ILP) formulation.
- Finally we provide experiments that show feasibility of our approaches, even for large AFs, based on instances of recent competitions.

2. Argumentation Frameworks

We recall basics of argumentation frameworks (AFs) [14] and their semantics (see also [2] for an introduction).

Definition 1. An *argumentation framework (AF)* is a pair $F = (A, R)$ where A is a finite [1] set of arguments and $R \subseteq A \times A$ is the attack relation. We say that $S \subseteq A$ *attacks* b if $(a,b) \in R$ for some $a \in S$. Moreover, an argument $a \in A$ is *defended* (in F) by $S \subseteq A$ if each b with $(b,a) \in R$ is attacked by S in F.

Furthermore we denote by $S^+ = \{b \in A \mid a \in S, (a,b) \in R\}$ the set of arguments attacked by S, and by $S^- = \{b \in A \mid a \in S, (b,a) \in R\}$ the set of arguments attacking an argument in S. We call $S \cup S^+$ the *range* of S in F.

Semantics for AFs are defined as functions σ which assign to each AF $F = (A,R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We consider for σ the functions *cf*, *adm*, *com*, *grd*, and *strAdm* which stand for conflict-free, admissible, complete, grounded, and strongly admissible extensions, respectively. We first recall some semantics already introduced by Dung [14].

Definition 2. Let $F = (A,R)$ be an AF. A set $S \subseteq A$ is *conflict-free (in F)*, if there are no $a,b \in S$, such that $(a,b) \in R$. By $cf(F)$ we denote the collection of conflict-free sets. For a conflict-free set $S \in cf(F)$, we say

- $S \in adm(F)$, if each $a \in S$ is defended by S;
- $S \in com(F)$, if $a \in S$ iff a is defended by S; and
- $S \in grd(F)$, if $S = \bigcap_{T \in com(F)} T$.

For each AF F we have $grd(F) \subseteq com(F) \subseteq adm(F) \subseteq cf(F)$ and $|grd(F)| = 1$, i.e. there is a unique grounded extension which is the $\subseteq$-minimal complete extension.

Next we introduce strongly admissible semantics as introduced by Baroni and Giacomin [4]. To this end, we first recall the notion of strong defence.

Definition 3. Let $F = (A,R)$ be an AF. An argument $a \in A$ is *strongly defended* by a set $S \subseteq A$ iff each $b \in \{a\}^-$ is attacked by some argument $c \in S \setminus \{a\}$ such that c is strongly defended by $S \setminus \{a\}$.

We are now ready to provide the definition of strongly admissible semantics.

Definition 4. Let $F = (A,R)$ be an AF. An $E \subseteq A$ is *strongly admissible* ($E \in strAdm(F)$) iff E strongly defends each of its arguments.

Caminada and Dunne [9] provide some useful characterizations of strongly admissible semantics. In particular, one characterization avoids the notion of strong defence but recursively refers to smaller strongly admissible sets.

Proposition 1 ([9])**.** *Let $F = (A,R)$ be an AF. It holds that $E \in strAdm(F)$ iff each $a \in E$ is defended by some strongly admissible set $S \subseteq E \setminus \{a\}$.*

Another useful characterization is based on the well-known (restricted) characteristic function of AFs, recalled next.

Definition 5. Given an AF $F = (A,R)$, the *characteristic function* $\mathscr{F}_F : 2^A \to 2^A$ of F is defined as $\mathscr{F}_F(S) = \{x \in A \mid S \text{ defends } x\}$. We will also consider the characteristic function restricted to a given set $E \subseteq A$: $\mathscr{F}_{F,E}(S) = \{a \in E \mid S \text{ defends } a\}$.

[1] Notice that the original definition is not limited to finite frameworks, but as we are studying computational properties we are only concerned with finite AFs.

By [14] it holds that the grounded extension of an AF F is the least fixed-point of the characteristic function $\mathscr{F}_F$. Caminada and Dunne [9] use the restricted variant of the characteristic function to characterize strongly admissible sets.

Proposition 2 ([9]). *Let $F = (A,R)$ be an AF. We have $E \in strAdm(F)$ iff E is the least fixed-point of $\mathscr{F}_{F,E}(.)$.*

We next recall useful properties of strongly admissible semantics [4,9]. For each AF F we have $grd(F) \subseteq strAdm(F) \subseteq adm(F) \subseteq cf(F)$. Moreover, $strAdm(F)$ forms a lattice, with the grounded extension as the top element and the empty set as the bottom element. That is, the grounded extension acts as the top element of the $strAdm(F)$-lattice as well as the bottom element of $com(F)$-semi-lattice. This yields the observation that $grd(F) = strAdm(F) \cap com(F)$, which we will use later on.

Lemma 1. *Let $F = (A,R)$ be an AF. It holds that $S \in grd(F)$ iff $S \in strAdm(F) \cap com(F)$.*

Proof. First, by the above the grounded extension is both strongly admissible and complete. Next, recall that the grounded extension is the unique minimal complete extension and the unique maximal strongly admissible set. That is, each strongly admissible set different from the grounded extension is not complete and each complete extension different from the grounded extension is not strongly admissible. □

3. Complexity Results

In this section we recap existing complexity result for strong admissibility and complement them by P-hardness results as well as by studying the problem of computing a minimum sized strongly admissible set for a given argument.

3.1. Standard Reasoning Problems

The standard problems in abstract argumentation (cf. [16]) are: Credulous acceptance $Cred_\sigma$, deciding whether a given argument is in at least one σ-extension; Skeptical acceptance $Skept_\sigma$, deciding whether a given argument is in all σ-extensions; Verification Ver_σ, deciding whether a given set of arguments is a σ-extension; and Non-emptiness $Exists_\sigma^{\neg\emptyset}$, deciding whether the AF has a non-empty σ-extension.

First, credulous reasoning with strongly admissible semantics corresponds to credulous reasoning with grounded semantics [9] and is thus P-complete [17]. Moreover, as the empty-set is always strongly admissible no argument is skeptically accepted and the problem becomes trivial. The Non-emptiness problem again corresponds to the respective problem for grounded semantics and is in L. Next, as shown in [9], verifying a strongly admissible set is in P and we next show that it is also P-hard by relating it to verifying the grounded extension.

Lemma 2. *Verifying whether a given set is strongly admissible is* P*-complete.*

Proof. As shown in [9] we can verify a strongly admissible set E in P by computing the least fixed-point of $\mathscr{F}_{F,E}(.)$. We know that verifying the grounded extension is P-complete [17] and verifying a complete extension is in L [16]. A logspace reduction from

verifying the grounded extension G of an AF F to verifying a strongly admissible set E of an AF F' simply tests whether G is complete. If not it returns a no instance (e.g., the AF $F' = (\{a\}, \{(a,a)\})$ and $E = \{a\}$), otherwise it returns the unmodified AF and extension, i.e., $F' = F$ and $E = G$. Then, by Lemma 1, it holds that G is the grounded extension of F iff E is strongly admissible in F'. □

3.2. Minimum size strongly admissible sets

Arguably, the standard reasoning problems fail to fully characterize the complexity of strongly admissible semantics as both credulous and skeptical acceptance can be solved without referring to the strongly admissible sets of the AF, i.e., only the empty-set and the grounded extension are used. In that light, and motivated by the usage of strongly admissible sets as justifications in grounded discussion games [9] we are interested in the problem of computing a minimum size strongly admissible set containing a given argument. The decision version of this problem is the k-Witness problem $k\text{-}Witness_\sigma$, deciding whether a given argument is in at least one σ-extension of size at most k. We remark that k is part of the input of this problem, but we keep the "k" in the problem name to emphasize the size constraint. We next show that $k\text{-}Witness_{strAdm}$ is NP-complete, which implies that there is no polynomial time algorithm that computes a minimum size strongly admissible set (unless P = NP).

Theorem 1. *k-$Witness_{strAdm}$ is NP-complete.*

Proof. For membership, non-deterministically construct a subset of the arguments and verify whether this set (i) contains the queried argument, (ii) contains at most k many arguments (for a given integer k), and (iii) is strongly admissible in the given AF. The last check can be done in polynomial time (see Lemma 2).

For hardness, we reduce from the NP-complete problem of deciding whether a given Boolean formula is satisfiable. Given a Boolean formula $\varphi = c_1 \wedge \cdots \wedge c_n$ in conjunctive normal form (CNF) over variables X with clause set C, construct AF $F_\varphi = (A, R)$ with $A = X \cup \overline{X} \cup C \cup D \cup \{\varphi\}$ and

$$\begin{aligned} R = &\{(c_i, \varphi) \mid c_i \in C\} \cup \{(d_x, \varphi) \mid d_x \in D\} \cup \\ &\{(x, c_i) \mid x \in c_i\} \cup \{(\overline{x}, c_i) \mid \neg x \in c_i\} \cup \\ &\{(x, d_x), (\overline{x}, d_x) \mid x \in X\} \end{aligned}$$

with $D = \{d_x \mid x \in X\}$. It follows that F_φ can be constructed in polynomial time for a given φ. An illustration of the reduction for an example formula is shown in Figure 2. We claim that φ is satisfiable iff there is an $E \in strAdm(F_\varphi)$ with (i) $\varphi \in E$ and (ii) $|E| \leq |X| + 1$. First assume that φ is satisfiable and let M be a model of φ. Consider the set $E = M \cup \{\bar{x} \in \bar{X} \mid x \notin M\} \cup \{\varphi\}$ of arguments. Clearly $|E| = |X| + 1$ and it remains to show that $E \in strAdm(F_\varphi)$. First we have $E \setminus \{\varphi\} \subseteq \mathscr{F}_{F,E}(\emptyset)$ as these arguments are not attacked at all. Moreover, by the assumption that M is a model it follows that all c_i and d_x are attacked by $E \setminus \{\varphi\}$ and thus φ is defended and thus $E = \mathscr{F}^2_{F,E}(\emptyset)$. We obtain that $E \in strAdm(F_\varphi)$.

Now assume $E \in strAdm(F_\varphi)$ with (i) $\varphi \in E$ and (ii) $|E| \leq |X| + 1$. In particular $E \in adm(F_\varphi)$. As $\varphi \in E$ we have $C \cup D \subseteq E^+$. By the arguments D we have for each

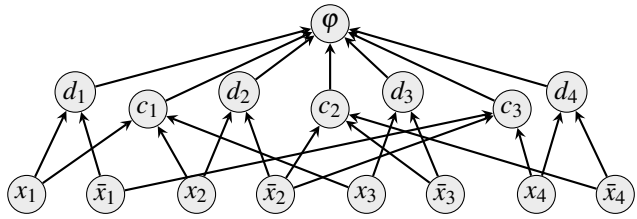

Figure 2. Example reduction for $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$.

$x \in X$ either $x \in E$ or $\bar{x} \in E$ and by the size constraint that not both of them are in E. As all C are attacked by E we obtain that $M = E \cap X$ is a model of φ. □

We summarize the complexity of all decision problems in Table 1.

Table 1. Computational complexity of strong admissibility

$Cred_{strAdm}$	$Skept_{strAdm}$	Ver_{strAdm}	$Exists^{\neg\emptyset}_{strAdm}$	$k\text{-}Witness_{strAdm}$
P-c	trivial	P-c	in L	NP-c

Given that we cannot compute a strongly admissible set of minimum size in polynomial time a standard approach would be to go for a strongly admissible sets whose size is a good approximation of the minimum size. We say a set S is an approximation within a factor α if we have $|S| \leq \alpha \cdot |opt|$ where opt is an optimal solution. An α-approximation algorithm is then a polynomial time algorithm that always returns a solution that is within a factor α.

In order to show that hardness even holds when approximating a strongly admissible set of minimum size with a queried argument, i.e., that under complexity theoretic assumptions there cannot be a c-approximation algorithm for this problem for any constant c, we consider the SET COVER problem. In the following we use $[n]$ as shorthand for the set $\{1,2,\ldots,n\}$ (for a positive integer n).

Definition 6 (SET COVER)**.** Given a universe $\mathtt{U} = [n]$ and a collection $\mathtt{S} = \{S_1,\ldots,S_m\}$ with $S_i \subseteq \mathtt{U}$, the SET COVER problem is to find a smallest set $I \subseteq [m]$ such that $\bigcup_{i\in I} S_i = \mathtt{U}$.

Notice that SET COVER is not a decision problem as we are interested in computing (the size of) a cardinality minimum solution. For SET COVER it is well-known that there is no α-approximation algorithm where α is a constant unless P = NP. The actual lower bound for approximation algorithms is even stronger.

Proposition 3 ([13])**.** *Approximating* SET COVER *within a factor* $(1-\varepsilon)\cdot\ln(n)$ *is* NP*-hard for every* $\varepsilon > 0$.

We next present a reduction from SET COVER to computing a minimum size strongly admissible set for a given argument.

Reduction 1. *For an instance* $(\mathtt{U},\mathtt{S})$ *of* SET COVER *we define the AF* $F_{\mathtt{U},\mathtt{S}} = (A,R)$ *with* $A = \mathtt{U} \cup \mathtt{S} \cup \{t\}$ *and* $R = \{(i,t) \mid i \in \mathtt{U}\} \cup \{(S,i) \mid S \in \mathtt{S}, i \in S\}$.

An example instance of this reduction is shown in Figure 3. We next show that this reduction maintains the size of minimum solutions.

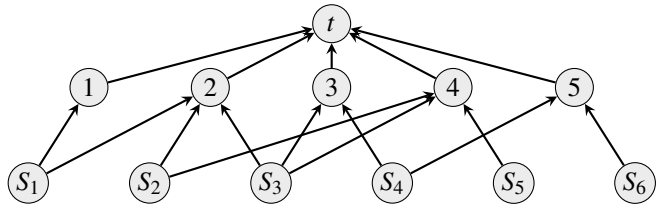

Figure 3. $F_{\mathtt{U},\mathtt{S}}$ with $\mathtt{U} = \{1,2,3,4,5\}$ and $\mathtt{S} = \{\{1,2\},\{2,4\},\{2,3,4\},\{3,5\},\{4\},\{5\}\}$.

Lemma 3. *Let $I_{\min}$ be a minimum set cover of $\mathtt{U},\mathtt{S}$ and E a minimum among the sets in $strAdm(F_{\mathtt{U},\mathtt{S}})$ containing t then $|I_{\min}|+1 = |E|$.*

Proof. We show that there is a one-to-one correspondence between set covers and strongly admissible sets containing t which maintains the size of the solutions. First consider a set cover $I \subseteq [m]$. It is easy to verify that the set $E = \{S_i \mid i \in I\} \cup \{t\}$ is a strongly admissible set in $F_{\mathtt{U},\mathtt{S}}$ and the $|I|+1 = |E|$, as by assumption the selected S_i attack all arguments in $\mathtt{U}$. Now consider $E \in strAdm(F_{\mathtt{U},\mathtt{S}})$ with $t \in E$ and define $I = \{i \in [m] \mid S_i \in E\}$. First as $t \in E$ we have $\mathtt{U} \cap E = \emptyset$. We thus have $\{S_i | i \in I\} = E \setminus \{t\} \subseteq [m]$ and $|I|+1 = |E|$. Finally, as the arguments $\mathtt{U}$ are only attacked by arguments $\mathtt{S}$ we have that each $i \in \mathtt{U}$ is contained in some $S \in E \cap \mathtt{S}$ and thus I is a set cover. □

By Lemma 3, each c-approximation algorithm for computing a minimum size strongly admissible set would yield a $(2c)$-approximation[2] for SET COVER, which is in contradiction to Proposition 3.

Theorem 2. *Computing a c-approximation for the minimum size of a strongly admissible set for a given argument is* NP*-hard for every $c \geq 1$.*

Let us complete this section on the computational complexity with some final remarks. First, notice that Theorem 2 implies Theorem 1. We believe that the hardness proof in terms of the standard reduction is of additional value (e.g., when comparing with other semantics) and thus included both reductions. Second, while we focused on minimizing the size of the strongly admissible set, all the results can be easily extended to minimizing the number of attackers of a strongly admissible set or to minimizing a (weighted) combination of the size and the number of attackers. Finally, notice that the AFs constructed in the reductions have a rather simple graph structure, i.e., they are acyclic, bipartite and all paths are of length at most 2.

4. Two Reduction-based Implementations

As our complexity analysis shows NP-hardness for computing strongly admissible sets of minimum size, we implement computation of strongly admissible sets via using answer set programming (ASP) and integer linear programming (ILP), two approaches that showed promise for NP-hard problems in computational argumentation [11,12]. Both approaches make use of the characterization as least fixed point of the characteristic function from Proposition 2.

[2] Assume there is a c-approximation, i.e., a strongly admissible set E' with $|E'| \leq c \cdot |E|$. Then, by Lemma 3, there is set cover I with $|I| = |E'|-1$ and thus we have $|I| = |E'|-1 \leq c \cdot |E|-1 = c \cdot (|I_{\min}|+1)-1 \leq 2c \cdot |I_{\min}|$.

4.1. Answer Set Programming Encodings

Background on ASP. We recall briefly ASP background [24,21]. We fix a countable set U of constants. An atom is an expression $p(t_1,\ldots,t_n)$, where p is a predicate of arity $n \geq 0$ and each term t_i is either a variable or an element from U. An atom is ground if it is free of variables. BU denotes the set of all ground atoms over U. A rule r is of the form

$$a \leftarrow b_1,\ldots,b_k,\ \texttt{not}\ b_{k+1},\ldots,\ \texttt{not}\ b_m.$$

with $m \geq k \geq 0$, where $a,b_1,\ldots,b_m$ are atoms, and "not" stands for default negation. The head of r is a and the body of r is $body(r) = \{b_1,\ldots,b_k,\ \texttt{not}\ b_{k+1},\ldots,\ \texttt{not}\ b_m\}$. Furthermore, $body^+(r) = \{b_1,\ldots,b_k\}$ and $body^-(r) = \{b_{k+1},\ldots,b_m\}$. A rule r is ground if r does not contain variables. A program is a finite set of rules. If each rule in a program is ground, we call the program ground.

For any program π, let UP be the set of all constants appearing in π. Define GP as the set of rules r_τ obtained by applying, to each rule $r \in \pi$, all possible substitutions τ from the variables in r to elements of UP. An interpretation $I \subseteq BU$ satisfies a ground rule r iff the head a of r is in I whenever $body^+(r) \subseteq I$ and $body^-(r) \cap I = \emptyset$. I satisfies a ground program π, if each $r \in \pi$ is satisfied by I. A non-ground rule r (resp., a program π) is satisfied by an interpretation I iff I satisfies all groundings of r (resp., GP). An interpretation $I \subseteq BU$ is an answer set of π if it is a subset-minimal set satisfying the Gelfond-Lifschitz reduct $\pi^I = \{head(r) \leftarrow body^+(r) \mid I \cap body^-(r) = \emptyset, r \in GP\}$.

ASP Encoding. As usual [18], we encode an AF $F = (A,R)$ as ASP facts $\{\mathbf{arg}(x) \mid x \in A\}$ and $\{\mathbf{att}(x,y) \mid (x,y) \in R\}$. We provide our ASP encoding for strongly admissible semantics in Listing 1. The first two lines generate a potential answer set for each subset E of the arguments, where the atoms with the **in** predicate contain the arguments in E. Lines 3 & 4 compute the least fixed-point of $\mathscr{F}_{F,E}(.)$, notice that in Line 3 we explicitly ensure that only arguments a with $\mathbf{in}(a)$ can be in the fixed-point. The conditional $\mathbf{defeated}(Y) : \mathbf{att}(Y,X)$ stands for a conjunction (list) of all $\mathbf{defeated}(Y)$ s.t. $\mathbf{att}(Y,X)$ holds (i.e., the conditional is expanded to $\{\mathbf{defeated}(y) \mid (y,x) \in R\}$). Finally, in Line 5 we rule out answer sets where the least fixed-point differs from the guessed set E. With the encoding in Listing 1 we can use clingo [20] to compute all strongly admissible

Listing 1: Encoding π_{strAdm}

```
in(X) ← arg(X), not out(X).
out(X)← arg(X), not in(X).
fixedPoint(X) ← in(X), defeated(Y) : att(Y,X).
defeated(X)← arg(X), fixedPoint(Y), att(Y,X).
← in(X), not fixedPoint(X).
```

sets of an AF and to solve all the standard reasoning tasks. Moreover, clingo also provides flexible optimization statements. To compute an optimal strongly admissible set that contains some argument t we first add a constraint "$\leftarrow$ not $\mathbf{in}(t)$." to ensure that the

computed set contains the argument and then add optimization constrains. To compute a minimum size set we add the constraint "`#minimize {1@1,X:in(X)}.`" which for each argument in the extension adds one to the objective function that is minimized.

If we want to take also the attackers of an extension into account we first add a rule "**attacker**`(X):-` **in**`(Y),` **att**`(X,Y).`" that computes the attacking arguments and then we can formulate minimize statements that also take attackers into account. For example such statements can minimize the size of the set plus the number of attackers(`#minimize {1@1,X:`**attacker**`(X)}. #minimize {1@1,X:`**in**`(X)}.`), among the minimum size sets minimize the number of attackers (`#minimize {1@2,X:`**in**`(X)}. #minimize {1@1,X:`**attacker**`(X)}.`), or weight between size and the number of attackers, e.g., by adding two for each argument in the set but just one for attackers (`#minimize {2@1,X:`**in**`(X)}. #minimize {1@1,X:`**attacker**`(X)}.`).

The encodings are available at `https://www.dbai.tuwien.ac.at/research/argumentation/aspartix/dung/min_extensions.html`.

4.2. Encoding as Integer Linear Programming

We describe an encoding of our problem as an Integer Linear Program (ILP) (see, e.g., [27]). Integer linear programming is a well-known NP-hard problem where one is given variables over the integer domain, a linear objective function and linear constraints, and one has to minimize (or maximize) the objective while satisfying the constraints.

In contrast to the ASP encoding we will require a quadratic number of variables and as preliminary tests showed that solvers are sensitive to the number of variables we implemented the following simplifications before encoding the problem as ILP. First we ignore all arguments that cannot reach the query argument t, second compute the grounded extension G of the simplified AF, and then give an encoding that only refers to arguments in G and G^-. Moreover, as our encoding mimics the (restricted) characteristic function we are also interested in the maximal number of iterations k until a fixed point is reached. We obtain that the number of iterations is at most $\min(|G|, |G^-|+1)$ as in each iteration we have to add an additional argument to G and attack an additional argument in G^-, otherwise we have reached a fixed-point.

Given an AF $F = (A,R)$, the grounded extension G, the attackers G^-, $k = \min(|G|, |G^-|+1)$, w_a, w_b coefficients to weight between $|E|$ and $|E|^-$, and a target argument $t \in A$ we define variables with domain $\{0,1\}$: $x_{i,\ell}$ encoding that argument $i \in G$ is accepted in the ℓ-th iteration of the fixed-point computation; and y_i encoding that $i \in G^-$ is an argument that attacks E. The ILP is then given as follows:

$$\min \quad w_a \cdot \sum_{i \in G} x_{i,\ell} + w_b \cdot \sum_{i \in G^-} y_i \tag{1}$$

$$x_{i,\ell} \leq x_{i,\ell+1} \qquad \forall i \in G, 1 \leq \ell < k \tag{2}$$

$$x_{j,\ell} \leq \sum_{(k,i) \in R} x_{k,\ell-1} \qquad \forall (i,j) \in R, 2 \leq \ell \leq k \tag{3}$$

$$x_{j,1} \leq 0 \qquad \forall (i,j) \in R \tag{4}$$

$$x_{j,k} \leq y_i \qquad \forall (i,j) \in R \tag{5}$$

$$x_t = 1 \tag{6}$$

In the objective function (1) we can use the parameters to specify if we want to minimize the arguments in the extension ($w_a = 1, w_b = 0$), the number of attackers of the extension ($w_a = 0, w_b = 1$), or the sum of both ($w_a = w_b = 1$). The constraint (2) ensures that if an argument is accepted in the ℓ-th iteration then it is also accepted in all the later iterations. By constraint (3) we get that an argument is accepted in the ℓ-th iteration only if it is defended by the arguments accepted at the $(\ell - 1)$-th iteration. With the exception of the first iteration where constraint (4) ensures that only unattacked arguments are accepted. Constraint (5) encodes that all arguments i that attack an accepted argument are marked as attackers of E. Finally, constraint (6) ensures that the computed strongly admissible set contains the query argument t.

5. Experimental Evaluation

We provide an empirical evaluation of our two reduction-based approaches to implement strongly admissible sets in ASP and ILP. We focus on the task of finding one strongly admissible set of minimum size that contains a queried argument for a given AF, for which we showed NP-hardness.

For instances, we considered AFs and queries provided by the benchmark sets of the two most recent argumentation competitions ICCMA'17 [19] and ICCMA'19[3]. From ICCMA'19 we considered all provided AFs and queries, and from ICCMA'17 we considered the AFs and queries from the "A" benchmark set. From these benchmark sets, we included in our experiments all AFs and queries whenever a query was provided by the competition. Additionally, for each AF we generated one query argument within the grounded extension of the AF (whenever the grounded is not empty). This resulted in 326 AFs from ICCMA'19 and 333 AFs from ICCMA'17 (17 AFs from ICCMA'17 included no query argument and have an empty grounded extension). Furthermore, to look at scalability, we generated 22 new AFs from the admbuster class [7], which is specifically designed for strongly admissible sets, with sizes from $10,000$ to $7,000,000$ arguments. For queries of the newly generated AFs, we included again one randomly chosen argument from the grounded extension, and also the distinguished argument "a", which requires the whole grounded extension to be included (i.e., the only strongly admissible set containing "a" is the grounded extension). Overall, we included 698 AFs and 1168 queries over these AFs.

We let clingo [20] v5.4.0 and IBM's CPLEX [1] v12.10.0.0 compute a strongly admissible set of minimum size containing the queried argument with a timeout limit of 900 seconds and a memory limit of 8GB per query. All experiments were run on a machine with two AMD Opteron Processors 6308, 12 x 16GB RAM, and Debian 8. For using CPLEX, we used the python LP modeler PuLP [4] to generate the ILP constraints.

We summarize the results obtained. Using clingo and the above encoding, 1157 instances were solved optimally (550) or clingo reported unsatisfiability (607). One timeout was encountered and ten times the memory limit was reached (for instances with at least $3,000,000$ arguments). Using CPLEX, overall 1089 instances were solved, either by reporting an optimal strongly admissible set (482) or by showing unsatisfiability (607). Further, using CPLEX two timeouts were reported and 76 times the memory

[3] `https://www.iccma2019.dmi.unipg.it/`

[4] https://pypi.org/project/PuLP/

Table 2. Summary of performance evaluation

approach	# optima found	# unsatisfiability reported	# timeouts	# memory limit reached
ASP	550	607	1	10
ILP	482	607	2	77

limit was reached. One time a memory error was reported. Considering the running times clingo solved 75% of the instances within 1.6 sec while CPLEX solved 75% of the instances within 2.5 sec. When considering the admbuster instances, clingo solved all instances up to $2,000,000$ arguments while CPLEX only solved some of the instances up to $20,000$ arguments. In Table 2 we summarize the results obtained (the memory error is included in the memory limit reached column).

From the results one can conclude that a large portion of the instances could be solved (optimally), even when faced with large and potentially complex AFs. Due to the low number of timeouts, we hypothesize that memory was the main limiting factor, for the instances considered. Both reduction-based approaches reported the same unsatisfiable instances, which plausibly seems to be a simple case: computing the grounded extension and checking inclusion is a poly-time decidable problem. While our approach utilizing CPLEX reported a higher number of cases where the memory limit was exceeded, we speculate that this is more inherent to the large number of constraints produced during construction of the ILP rather than due to (limitations of) CPLEX itself. More efficient constructions of constraints might lead to better performance. Nevertheless, both approaches solved a majority of the instances.

6. Conclusions

In this paper we studied the computational properties of strongly admissible sets. Concretely, we showed NP-hardness of finding a minimum-sized strongly admissible set containing a queried argument, a hardness result that we showed also to hold when approximating strongly admissible sets. To overcome the clear theoretic complexity barrier, we provided two approaches to compute strongly admissible sets in practice: based on the promising approaches of ASP and ILP, we provided one implementation each, with both of them showing good performance in our experiments. The implementation based on ASP was somewhat outperforming the approach based on ILP.

Directions for future work include extending our approaches to minimal admissible sets, which are also relevant for discussion-games [10], and abstract argumentation formalisms that enhance Dung AFs [22].

Acknowledgements

This work was supported by the Austrian Science Fund (FWF) through project P30168-N31 and the Vienna Science and Technology Fund (WWTF) through project ICT19-065.

References

[1] IBM ILOG: CPLEX optimizer 12.10.0.0, 2020. Webpage at IBM: `https://www.ibm.com/analytics/cplex-optimizer`.

[2] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.
[3] Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors. *Handbook of Formal Argumentation*. College Publications, 2018.
[4] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artif. Intell.*, 171(10-15):675–700, 2007.
[5] Ringo Baumann, Thomas Linsbichler, and Stefan Woltran. Verifiability of argumentation semantics. In *Proc. COMMA*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 83–94. IOS Press, 2016.
[6] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artif. Intell.*, 171(10-15):619–641, 2007.
[7] Martin Caminada. Strong admissibility revisited. In *Proc. COMMA*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 197–208. IOS Press, 2014.
[8] Martin Caminada. A discussion game for grounded semantics. In *Proc. TAFA, Revised Selected Papers*, volume 9524 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2015.
[9] Martin Caminada and Paul E. Dunne. Strong admissibility revisited: Theory and applications. *Argument & Computation*, 10(3):277–300, 2019.
[10] Martin W. A. Caminada, Wolfgang Dvořák, and Srdjan Vesic. Preferred semantics as socratic discussion. *J. Log. Comput.*, 26(4):1257–1292, 2016.
[11] Federico Cerutti, Sarah A. Gaggl, Matthias Thimm, and Johannes P. Wallner. Foundations of implementations for formal argumentation. In *Handbook of Formal Argumentation*, chapter 15, pages 688–767. College Publications, 2018.
[12] Günther Charwat, Wolfgang Dvořák, Sarah A. Gaggl, Johannes P. Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation – A survey. *Artif. Intell.*, 220:28–63, 2015.
[13] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proc. STOC*, pages 624–633. ACM, 2014.
[14] Phan Minh Dung. 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.
[15] Paul E. Dunne. Characterizing strongly admissible sets. *Argument & Computation*, 2020. Accepted manuscript available at `http://dx.doi.org/10.3233/AAC-200483`.
[16] Wolfgang Dvořák and Paul E. Dunne. Computational problems in formal argumentation and their complexity. In *Handbook of Formal Argumentation*, chapter 13, pages 631–688. College Publications, 2018.
[17] Wolfgang Dvořák and Stefan Woltran. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res.*, 41:445–475, 2011.
[18] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.
[19] Sarah Alice Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. Design and results of the second international competition on computational models of argumentation. *Artif. Intell.*, 279, 2020.
[20] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Patrick Lühne, Philipp Obermeier, Max Ostrowski, Javier Romero, Torsten Schaub, Sebastian Schellhorn, and Philipp Wanko. The Potsdam answer set solving collection 5.0. *KI*, 32(2-3):181–182, 2018.
[21] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. ICLP/SLP*, pages 1070–1080. MIT Press, 1988.
[22] Atefeh Keshavarzi Zafarghandi, Rineke Verbrugge, and Bart Verheij. Discussion games for preferred semantics of abstract dialectical frameworks. In *Proc. ECSQARU*, volume 11726 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2019.
[23] Sanjay Modgil and Martin Caminada. Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009.
[24] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3-4):241–273, 1999.
[25] Henry Prakken and Giovanni Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1):25–75, 1997.
[26] Zeynep G. Saribatur, Johannes P. Wallner, and Stefan Woltran. Explaining non-acceptability in abstract argumentation. In *Proc. ECAI*, 2020. Accepted for publication.
[27] Gerard Sierksma and Yori Zwols. *Linear and integer optimization: theory and practice*. CRC Press, 2015.