An Efficient Algorithm for Admissible Argumentation Stages*

Tobia Zanetti[1], Massimiliano Giacomin[1], Mauro Vallati[2], and Federico Cerutti[1,3]

[1] Università degli Studi di Brescia
[2] University of Huddersfield
[3] Cardiff University

Abstract. In this paper we introduce AASExts, an algorithm for computing admissible argumentation stage extensions—a.k.a. semi–stable extensions. Admissible argumentation stage extensions play a decisive role in unifying two lines of research in formal argumentation: admissible-based extensions as suggested by Dung in his seminar paper; and the traditional approach based on dialectical evaluation of the defeat status of arguments. In this paper, we improve techniques developed for other semantics, notably preferred semantics, as well as leverage—for the first time—recent advances in All-SAT community. We prove our proposed algorithm is sound and complete, and we show empirically that our implementation significantly outperforms even sophisticated ASP-based and SAT-based reduction approaches on existing benchmarks.

Keywords: Abstract Argumentation; Semi-Stable Semantics; Algorithm.

1 Introduction

Research based on Dung's model of argumentation [13]—that considers only abstract arguments and attack relations between them—identified several *semantics*, viz. criteria for selecting *extensions*, i.e. sub-sets of arguments acceptable in some sense. Pivotal in Dung's theory is the notion of *admissible set*, i.e. conflict-free and defending itself against attacks. Building on top of such a notion, Dung [13] introduced the concept of grounded, stable, and preferred semantics. An interested reader is referred to Baroni *et al.* [2] for an introduction.

However, as noted by Rescher [25] among others, dialectical argumentation has two main characteristics. On the one side, arguments in favour of a conclusion can be challenged by other (counter)arguments. On the other side, the acceptability status of an argument—or of its conclusion—depends on the stage of the argumentation process, i.e. the process of supporting or opposing arguments. Following this intuition, the acceptability status of an argument can be either *undefeated* or *defeated* [29]. Any argument attacked by an undefeated argument should be defeated, and any defeated argument must be attacked by at least one undefeated argument: this idea will then be re-named as labelling [5, 7].

An argumentation stage [29], then, is a set of arguments whose acceptability status is either undefeated or defeated w.r.t. to the conditions above. Arguments whose acceptability status is neither undefeated nor defeated do not belong to such a stage. This naturally leads to a notion of argumentation stage extension as a maximum—w.r.t. set inclusion—argumentation stage. However, as shown in [30], this notion of argumentation stage does not guarantee the admissibility property proposed by Dung [13].

To unify these two lines of research, Verheij [30] proposed the notion of admissible argumentation stage extension as an argumentation stage with an admissible set of undefeated arguments: this notion will be then re-named as semi–stable extension [6, 7]. Semi-stable semantics has unique interesting properties, in particular it coincides with stable semantics in the case a stable extension exists; and each admissible stage extension is also a preferred extension.

Wallner *et al.* [31] proposed SSTMCS, an algorithm for computing admissible stage extensions exploiting algorithms for computing minimal correction sets (MCS) [20, 21], i.e. subset-minimal sets of clauses of a formula. argmat-sat [24] reimplemented a very similar idea and scored first during the 2017 edition of the International Competition on Computational Models of Argumentation. An alternative system, Aspartix [16], exploits answer set programming [23] for computing argumentation semantics extensions, including admissible stage extensions.[4]

In this paper, we improve recent advancements in exploiting SAT solvers as NP-oracles [14, 9, 11], and we propose AASExts, an algorithm for computing admissible argumentation stage extensions that reduces the problem of identifying argumentation stages to a SAT problem.[5] Our extensive experimental analysis supports the claim that our proposal despite its simplicity, performs better than some of the existing approaches looking at ASP-based and SAT-based reductions for computing admissible argumentation stage extensions. AASExts has been included in the version of the ArgSemSAT solver that took part in the 2017 edition of the ICCMA competition. The competition included a track focused on semi-stable extensions: ArgSemSAT achieved the second place of the track.[6] For a comparison with the other systems that participated in the 2017 edition, we refer the readers to [18]. We refrained from comparisons with the 2019 edition as we are aware—from personal communication—that the organisers are currently working on an extensive analysis also considering ArgSemSAT—that did not participate in the 2019 edition—as a baseline.

In order to adhere to the current terminological standards [2, 26], hereafter we will consider the alternative definitions provided in works from Caminada (*et al.*) [6, 5, 7], summarised in Section 2. In Section 3 we discuss the theoretical foundations of our proposal, AASExts, and in Section 4 the outcomes of our experimentation analysis. Finally, in Section 5 we draw the conclusions and discuss future avenues of research.

[4] http://www.dbai.tuwien.ac.at/research/project/argumentation/systempage/

[5] Implementation available at https://github.com/federicocerutti/ArgSemSAT.

[6] http://argumentationcompetition.org/2017/index.html

2 Dung's Argumentation Framework

An argumentation framework [13] consists of a set of arguments[7] and a binary attack relation between them.

Definition 1. *An* argumentation framework *(AF) is a pair $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$.*

We say that $\boldsymbol{b}$ attacks $\boldsymbol{a}$ *iff $\langle \boldsymbol{b}, \boldsymbol{a} \rangle \in \mathcal{R}$, also denoted as $\boldsymbol{b} \rightarrow \boldsymbol{a}$. The set of attackers of an argument $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^- \triangleq \{\boldsymbol{b} : \boldsymbol{b} \rightarrow \boldsymbol{a}\}$, the set of arguments attacked by $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^+ \triangleq \{\boldsymbol{b} : \boldsymbol{a} \rightarrow \boldsymbol{b}\}$.*

We also extend these notations to sets of arguments, i.e. given $E, S \subseteq \mathcal{A}$, $E \rightarrow \boldsymbol{a}$ iff $\exists \boldsymbol{b} \in E$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$; $\boldsymbol{a} \rightarrow E$ iff $\exists \boldsymbol{b} \in E$ s.t. $\boldsymbol{a} \rightarrow \boldsymbol{b}$; $E \rightarrow S$ iff $\exists \boldsymbol{b} \in E, \boldsymbol{a} \in S$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$; $E^- \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{b} \rightarrow \boldsymbol{a}\}$ and $E^+ \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{a} \rightarrow \boldsymbol{b}\}$.

The range of a set of arguments $S \subseteq \mathcal{A}$ is $S \cup S^+$.

Each argumentation framework has an associated directed graph where the vertices are the arguments, and the edges are the attacks.

The basic properties of conflict–freeness, acceptability, and admissibility of a set of arguments are fundamental for the definition of argumentation semantics.

Definition 2. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* conflict–free *set of Γ if $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ s.t. $\boldsymbol{a} \rightarrow \boldsymbol{b}$;*
- *an argument $\boldsymbol{a} \in \mathcal{A}$ is* acceptable *with respect to a set $S \subseteq \mathcal{A}$ of Γ if $\forall \boldsymbol{b} \in \mathcal{A}$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$, $\exists\, \boldsymbol{c} \in S$ s.t. $\boldsymbol{c} \rightarrow \boldsymbol{b}$;*
- *the function $\mathcal{F}_\Gamma : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ such that $\mathcal{F}_\Gamma(S) = \{\boldsymbol{a} \mid \boldsymbol{a}$ is acceptable w.r.t. $S\}$ is called the* characteristic function *of Γ;*
- *a set $S \subseteq \mathcal{A}$ is an* admissible *set of Γ if S is a conflict–free set of Γ and every element of S is acceptable with respect to S, i.e. $S \subseteq \mathcal{F}_\Gamma(S)$.*

An argumentation semantics σ prescribes for any AF Γ a set of *extensions*, denoted as $\mathcal{E}_\sigma(\Gamma)$, namely a set of sets of arguments satisfying the conditions dictated by σ. Here we need to recall the definitions of stable (denoted as ST), preferred (denoted as PR), and admissible argumentation stage or semi–stable (denoted as SST) semantics.

Definition 3. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* stable extension *of Γ, i.e. $S \in \mathcal{E}_{\mathsf{ST}}(\Gamma)$, iff S is a conflict-free set of Γ and $S \cup S^+ = \mathcal{A}$;*
- *a set $S \subseteq \mathcal{A}$ is a* preferred extension *of Γ, i.e. $S \in \mathcal{E}_{\mathsf{PR}}(\Gamma)$, iff S is a maximal (w.r.t. set inclusion) admissible set of Γ;*
- *a set $S \subseteq \mathcal{A}$ is a* semi–stable extension *of Γ, i.e. $S \in \mathcal{E}_{\mathsf{SST}}(\Gamma)$, iff S is an admissible set where $S \cup S^+$ (i.e. its range) is maximal (w.r.t. set inclusion).*

It is immediate to see that if a stable extension exists, the semi–stable extensions coincide with the stable extensions.

[7] In this paper we consider only *finite* sets of arguments: see Baroni *et al.* [3] for a discussion on infinite sets of arguments.

Proposition 1. *Given an AF* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$*, if* $\mathcal{E}_{\mathtt{ST}}(\Gamma) \neq \emptyset$*, then* $\mathcal{E}_{\mathtt{ST}}(\Gamma) = \mathcal{E}_{\mathtt{SST}}(\Gamma)$.

Proof. Immediate from Definitions 2 and 3.

The notion of complete extension has been introduced as an auxiliary definition [13]. Given an $AF\ \Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, a set $S \subseteq \mathcal{A}$ is a *complete extension* of Γ iff S is a conflict-free set of Γ and $S = \mathcal{F}_\Gamma(S)$.

Each extension S implicitly defines a three-valued *labelling* of arguments, or dialectical evaluation: an argument **a** is labelled `in` (*undefeated* [29]) iff $\mathbf{a} \in S$; is labelled `out` (*defeated* [29]) iff $\exists\ \mathbf{b} \in S$ s.t. $\mathbf{b} \rightarrow \mathbf{a}$; is labelled `undec` if neither of the above conditions holds. In the light of this correspondence, argumentation semantics can be equivalently defined in terms of labellings rather than of extensions [2].

Definition 4. *Given a set of arguments* S*, a* labelling *of* S *is a total function* $\mathcal{L}ab : S \mapsto \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$*. The set of all* labellings *of* S *is denoted as* $\mathfrak{L}_S$*. Given an* $AF\ \Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$*, a* labelling *of* Γ *is a labelling of* $\mathcal{A}$*. The set of all* labellings *of* Γ *is denoted as* $\mathfrak{L}(\Gamma)$.

Given a labelling $\mathcal{L}ab$, it is possible to write $\mathtt{in}(\mathcal{L}ab)$ for $\{A | \mathcal{L}ab(A) = \mathtt{in}\}$, $\mathtt{out}(\mathcal{L}ab)$ for $\{A | \mathcal{L}ab(A) = \mathtt{out}\}$ and $\mathtt{undec}(\mathcal{L}ab)$ for $\{A | \mathcal{L}ab(A) = \mathtt{undec}\}$.

Complete labellings can be defined as follows.

Definition 5. *Let* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. A labelling* $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ *is a* complete labelling *of* Γ *iff it satisfies the following conditions for any* $\boldsymbol{a} \in \mathcal{A}$*:*

- $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) = \mathtt{out}$*;*
- $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$*;*
- $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) \neq \mathtt{in} \wedge \exists \boldsymbol{c} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{c}) = \mathtt{undec}$.

The stable, preferred, and semi–stable labelling can then be defined on the basis of complete labellings.

Definition 6. *Let* $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. A labelling* $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ *is*

- *a* stable labelling *of* Γ *if it is a complete labelling of* Γ *and there is no argument labelled* `undec`*;*
- *a* preferred labelling *of* Γ *if it is a complete labelling of* Γ *maximising the set of arguments labelled* `in`*;*
- *a* semi–stable labelling *of* Γ *if it is a complete labelling of* Γ *minimising the set of arguments labelled* `undec`*;*

In order to show the connection between extensions and labellings, let us recall the definition of the function `Ext2Lab`,[2] returning the labelling corresponding to a conflict–free set of arguments S.

Definition 7. *Given an* $AF\ \Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ *and a conflict–free set* $S \subseteq \mathcal{A}$*, the corresponding labelling* $\mathtt{Ext2Lab}(S)$ *is defined as* $\mathtt{Ext2Lab}(S) \equiv \mathcal{L}ab$*, where*

- $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \boldsymbol{a} \in S$

– $\mathcal{L}ab(\boldsymbol{a}) = \texttt{out} \Leftrightarrow \exists\, \boldsymbol{b} \in S \textit{ s.t. } \boldsymbol{b} \rightarrow \boldsymbol{a}$
– $\mathcal{L}ab(\boldsymbol{a}) = \texttt{undec} \Leftrightarrow \boldsymbol{a} \notin S \wedge \nexists\, \boldsymbol{b} \in S \textit{ s.t. } \boldsymbol{b} \rightarrow \boldsymbol{a}$

Caminada [5] shows that there is a bijective correspondence between the complete, stable, preferred, and semi–stable extensions and the complete, stable, preferred, and semi–stable labellings, respectively.

Proposition 2. *Given an an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{L}ab$ is a complete (stable, preferred, semi–stable) labelling of Γ if and only if there is a complete (stable, preferred, semi–stable) extension S of Γ such that $\mathcal{L}ab = \texttt{Ext2Lab}(S)$.*

Proof. See Baroni et al. *[2].*

A propositional formula over a set of boolean variables is satisfiable iff there exists a truth assignment of the variables such that the formula evaluates to True. Checking whether such an assignment exists is the satisfiability (SAT) problem. Following Cerutti *et al.* [9] where the case of preferred semantics is considered, given an $AF\ \Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ we derive a boolean formula, called *complete labelling formula* and denoted as Π_Γ, such that each satisfying assignment of the formula corresponds to a complete labelling. For each argument $\mathbf{a} \in \mathcal{A}$ we define three boolean variables, $I_\mathbf{a}$, $O_\mathbf{a}$, and $U_\mathbf{a}$, with the intended meaning that $I_\mathbf{a}$ is true when argument $\mathbf{a}$ is labelled `in`, false otherwise, and analogously $O_\mathbf{a}$ and $U_\mathbf{a}$ correspond to labels `out` and `undec`. Formally, given $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ we define the corresponding set of variables as $\mathcal{V}(\Gamma) \triangleq \cup_{\mathbf{a} \in \mathcal{A}} \{I_\mathbf{a}, O_\mathbf{a}, U_\mathbf{a}\}$. Now we express the constraints of Definition 5 in terms of the variables $\mathcal{V}(\Gamma)$.

We reuse the same encoding that Cerutti *et al.* [11] has shown to have best performance in the case of enumerating preferred extensions, namely:

$$
\begin{aligned}
\Pi_\Gamma = & \bigwedge_{\mathbf{a} \in \mathcal{A}} \Big((I_\mathbf{a} \vee O_\mathbf{a} \vee U_\mathbf{a}) \wedge (\neg I_\mathbf{a} \vee \neg O_\mathbf{a}) \wedge \\
& \qquad (\neg I_\mathbf{a} \vee \neg U_\mathbf{a}) \wedge (\neg O_\mathbf{a} \vee \neg U_\mathbf{a}) \Big) \\
& \wedge \bigwedge_{\{\mathbf{a} \mid \mathbf{a}^- = \varnothing\}} I_\mathbf{a} \wedge \neg O_\mathbf{a} \wedge \neg U_\mathbf{a} \\
& \wedge \bigwedge_{\mathbf{a} \in \mathcal{A}} \bigwedge_{\{\mathbf{b} \mid \mathbf{b} \rightarrow \mathbf{a}\}} \neg I_\mathbf{a} \vee O_\mathbf{b} \\
& \wedge \bigwedge_{\mathbf{a} \in \mathcal{A}} \neg O_\mathbf{a} \vee \bigvee_{\{\mathbf{b} \mid \mathbf{b} \rightarrow \mathbf{a}\}} I_\mathbf{b} \\
& \wedge \bigwedge_{\mathbf{a} \in \mathcal{A}} \bigwedge_{\{\mathbf{b} \mid \mathbf{b} \rightarrow \mathbf{a}\}} \neg U_\mathbf{a} \vee \neg I_\mathbf{b} \\
& \wedge \bigwedge_{\mathbf{a} \in \mathcal{A}} \neg U_\mathbf{a} \vee \bigvee_{\{\mathbf{b} \mid \mathbf{b} \rightarrow \mathbf{a}\}} U_\mathbf{b}
\end{aligned}
$$

Π_Γ encodes in CNF the conditions $\mathcal{L}ab(\mathbf{a}) = \texttt{in} \Rightarrow \forall \mathbf{b} \in \mathbf{a}^- \mathcal{L}ab(\mathbf{b}) = \texttt{out}$; $\mathcal{L}ab(\mathbf{a}) = \texttt{out} \Rightarrow \exists \mathbf{b} \in \mathbf{a}^- : \mathcal{L}ab(\mathbf{b}) = \texttt{in}$; $\mathcal{L}ab(\mathbf{a}) = \texttt{undec} \Rightarrow \forall \mathbf{b} \in \mathbf{a}^- \mathcal{L}ab(\mathbf{b}) \neq \texttt{in} \wedge \exists \mathbf{c} \in \mathbf{a}^- : \mathcal{L}ab(\mathbf{c}) = \texttt{undec}$; together with the requirement of it being a total function.

3 Overview of AASExts

In this section we introduce AASExts, our proposal for computing semi–stable extensions. To this aim, let us first consider the following intermediate theoretical results.

Firstly, to strictly expand the range of a complete extension—in order to minimise the set of undecided arguments given a complete labelling $\mathcal{L}ab$—it is necessary to transform a label from `undec` into `in` or `out`. However, no constraints should be imposed on the arguments labelled either `in` or `out` in $\mathcal{L}ab$. Those arguments are *free* to change their labels, provided that they do not become `undec`.

Lemma 1. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ a complete labelling.*

$\forall \mathcal{L}ab' \in \mathfrak{L}(\Gamma)$ such that $\texttt{undec}(\mathcal{L}ab) \supset \texttt{undec}(\mathcal{L}ab')$ $\exists \boldsymbol{a} \in \mathcal{A}$ such that $\mathcal{L}ab(\boldsymbol{a}) =$ `undec` and $\mathcal{L}ab'(\boldsymbol{a}) = \{\texttt{in}, \texttt{out}\}$, and $\nexists \boldsymbol{b} \in \mathcal{A}$ such that $\mathcal{L}ab(\boldsymbol{b}) \neq$ `undec` and $\mathcal{L}ab'(\boldsymbol{b}) =$ `undec`.

Proof. Immediate from Definition 5.

Secondly, given the *freedom* of argument labelled `in` or `out` to swap their labels mentioned above, there might be multiple semi–stable labellings having the same set of `undec` arguments: they differ on the basis of the labels of the remaining arguments labelled either `in` or `out`.

Lemma 2. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ a semi–stable labelling.*

Then $\{\mathcal{L}ab' | \mathcal{L}ab'$ is semi–stable and $\texttt{undec}(\mathcal{L}ab') = \texttt{undec}(\mathcal{L}ab)\} = \{\mathcal{L}ab' | \mathcal{L}ab'$ is complete and $\texttt{undec}(\mathcal{L}ab') = \texttt{undec}(\mathcal{L}ab)\}$.

Proof. First, semi–stable labellings are complete by Definition 6. On the other hand, given a complete labelling $\mathcal{L}ab'$ such that $\texttt{undec}(\mathcal{L}ab') = \texttt{undec}(\mathcal{L}ab)$, $\texttt{undec}(\mathcal{L}ab')$ is minimal since $\mathcal{L}ab$ is semi–stable, thus $\mathcal{L}ab'$ is semi–stable.

AASExts resorts to several external functions: STExts, SS, I_A, U_A, and ALLSS. STExts is an algorithm for computing stable extensions. For the sake of completeness, Algorithm 1 shows a straightforward implementation of STExts: all the complete labellings with no `undec` arguments are enumerated at line 2 and their `in` arguments form stable extensions, cf. Definitions 6 and 7, and Proposition 2.

SS is a SAT solver—in this paper we used MiniSAT [15]—able to prove unsatisfiability too: it accepts as input a CNF formula and returns a variable assignment satisfying the formula if it exists, ε otherwise. I_A (resp. U_A) accepts as input a variable assignment concerning $\mathcal{V}(\Gamma)$ and returns the corresponding set of arguments labelled as `in` (resp. `undec`).

ALLSS is a solver for the All-SAT problem: in this paper we used the proposal illustrated in [34]. The All-SAT problem [22] deals with determining all the satisfying assignments that exist for a given propositional logic formula. A typical All-SAT solver is based on iteratively computing satisfying assignments using a traditional Boolean satisfiability (SAT) solver and adding blocking clauses which are the complement of the total/partial assignments.

Algorithm 1 STExts

Input: $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$
Output: $\mathcal{E}_{\mathsf{ST}}(\Gamma) \subseteq 2^{\mathcal{A}}$
1: $\mathcal{E}_{\mathsf{ST}}(\Gamma) := \varnothing$
2: **for each** $st \in \mathsf{ALLSS}\left(\Pi_\Gamma \wedge \bigwedge_{\mathbf{a} \in \mathcal{A}} \neg U_{\mathbf{a}}\right)$ **do**
3: $\quad \mathcal{E}_{\mathsf{ST}}(\Gamma) := \mathcal{E}_{\mathsf{ST}}(\Gamma) \cup \{\mathsf{I\_A}(st)\}$
4: **end for**
5: **return** $\mathcal{E}_{\mathsf{ST}}(\Gamma)$

AASExts is presented in Algorithm 2. At first it checks whether stable extensions exist (l. 1–4): in that case $\mathcal{E}_{\mathsf{ST}}(\Gamma) = \mathcal{E}_{\mathsf{SST}}(\Gamma)$ (Proposition 1).

Otherwise, it enforces a disjunctive clause to find at least one argument labelled `in` (l. 5) and then (l. 9–16) it starts the process to find a complete labelling with minimal set of `undec` arguments (Lemma 1), i.e. a semi–stable labelling.

Then it enumerates (l. 19–23) all the semi–stable labellings that share the same set of `undec` arguments (Lemma 2) before searching for a new semi–stable labelling with a different set of `undec` arguments (Definition 6).

Theorem 1. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework:* $\mathsf{AASExts}(\Gamma) = \mathcal{E}_{\mathsf{SST}}(\Gamma)$.

Proof ((Sketched due to space constraints)). The proof is analogous to [11, Theorem 2], and it builds on top of Proposition 1, Lemmas 1 and 2, and Definition 6. From Proposition 1, if stable extensions exist, they also are semi–stable. Otherwise, a CEGAR-like [14] approach to minimise the set of `undec` *arguments is performed.*

To illustrate the algorithm, let us consider the following example evolving the one introduced by Verheij [30].

Example 1. Let $\Gamma_1 = \langle \mathcal{A}_1, \mathcal{R}_1 \rangle$ where $\mathcal{A}_1 = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$ and $\mathcal{R}_1 = \{\langle \mathbf{a}, \mathbf{b} \rangle, \langle \mathbf{b}, \mathbf{a} \rangle, \langle \mathbf{a}, \mathbf{c} \rangle, \langle \mathbf{c}, \mathbf{d} \rangle, \langle \mathbf{d}, \mathbf{e} \rangle, \langle \mathbf{e}, \mathbf{c} \rangle, \langle \mathbf{f}, \mathbf{f} \rangle\}$.

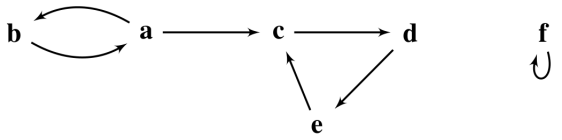

Fig. 1. Γ_1 as presented in Example 1

With reference to Example 1—depicted in Figure 1—let us suppose at l. 10 of Algorithm 2 the *compl* assignment identified is such that the corresponding labelling $\mathcal{L}ab_{compl} = \langle \mathtt{in}(\mathcal{L}ab_{compl}), \mathtt{out}(\mathcal{L}ab_{compl}), \mathtt{undec}(\mathcal{L}ab_{compl}) \rangle = \langle \{\mathbf{b}\}, \{\mathbf{a}\},$

$\{\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}\rangle$. Then l. 13 of Algorithm 1 enforces that arguments in $\texttt{in}(\mathcal{L}ab_{compl}) \cup \texttt{out}(\mathcal{L}ab_{compl}) = \{\mathbf{a}, \mathbf{b}\}$ can be labelled either `in` or `out`; and l. 14 requires that at least one argument belonging to $\texttt{undec}(\mathcal{L}ab_{compl}) = \{\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$ should be labelled either `in` or `out`.

During the second execution of the loop (l. 9–16), at l. 10 the only $compl'$ assignment that satisfies the additional constraints is such that $\mathcal{L}ab_{compl'} = \langle\{\mathbf{a}, \mathbf{d}\}, \{\mathbf{b}, \mathbf{c}, \mathbf{e}\}, \{\mathbf{f}\}\rangle$. Similarly as above, Algorithm 2 tries to label **f** either `in` or `out`, but at the third execution of the loop (l. 9–16) there is no further assignment able to satisfy such an additional constraint, therefore the loop is exited with $sstcand = compl'$ a variable assignment equivalent to a semi-stable labelling.

It is worth noticing that $\mathcal{E}_{\mathsf{ST}}(\Gamma_1) = \varnothing$ because **f** is self-defeating and it is isolated from the rest of the framework, therefore in this case $\mathcal{E}_{\mathsf{SST}}(\Gamma_1) \neq \mathcal{E}_{\mathsf{ST}}(\Gamma_1)$. However, if we restrict Γ_1 to the set of arguments $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}\}$, i.e. we *ignore* **f** and its self-defeating attack, then the stable and semi–stable extensions would coincide.

Finally, it also worth noticing that $\mathcal{E}_{\mathsf{SST}}(\Gamma_1) \neq \mathcal{E}_{\mathsf{PR}}(\Gamma_1)$. Indeed, there is another maximal admissible set of arguments, namely $\{\mathbf{b}\}$, i.e. a second preferred extension. However, its range—$\{\mathbf{b}, \mathbf{a}\}$—is not maximal.

4 Evaluation of AASExts

In this section, we present the result of a large experimental analysis comparing the performance of AASExts with respect to state-of-the-art approaches, on sets of differently-shaped AFs.

We implemented AASExts in C++. As per SS, we relied on MiniSAT [15], a small, complete, and efficient SAT-solver in the style of conflict-driven learning. Moreover, we considered the ALLSS developed by Yu *et al.* [34]. As mentioned above, a typical All-SAT solver is based on iteratively computing satisfying assignments using a traditional SAT solver and adding blocking clauses which are the complement of the total/partial assignments. Yu *et al.* [34] argue that such an algorithm is doing more work than needed and introduce more efficient algorithms: they also use MiniSAT as underlying SAT solver for their implementation.

4.1 Experimental Setup

We randomly generated 2,500 AFs based on five different graph models: Barabasi-Albert [1], Erdös-Rényi [17], Watts-Strogatz [32], graphs featuring a large number of stable extensions (hereinafter StableM), and a modified version of StableM (hereinafter SemiStableM) adding an artificial self-defeating attack detached from the rest of the graph—similarly to Example 1, cf. Figure 1—thus ensuring that no stable extension exists.

Erdös-Rényi graphs [17] are generated by randomly selecting attacks between arguments according to a uniform distribution. While Erdös-Rényi was the predominant model used for randomly generated experiments, [4] investigated also other graph structures such as *scale-free* and *small-world* networks. As discussed by Barabasi and Albert [1], a common property of many large networks is that the node connectivities follow a

Algorithm 2 AASExts

Input: $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$
Output: $\mathcal{E}_{\mathsf{SST}}(\Gamma) \subseteq 2^{\mathcal{A}}$

$\mathcal{E}_{\mathsf{SST}}(\Gamma) := \mathsf{STExts}(\langle \mathcal{A}, \mathcal{R} \rangle)$
if $\mathcal{E}_{\mathsf{SST}}(\Gamma) \neq \varnothing$ **then**
  return $\mathcal{E}_{\mathsf{SST}}(\Gamma)$
end if
$ocnf := \Pi_\Gamma \wedge \bigvee_{\mathbf{a} \in \mathcal{A}} I_{\mathbf{a}}$
repeat
  $icnf := ocnf$
  $sstcand := \varnothing$
  repeat
    $compl := \mathsf{SS}(icnf)$
    if $compl \neq \varepsilon$ **then**
      $sstcand := compl$
      $icnf := icnf \wedge \bigwedge_{\mathbf{a} \notin \mathsf{U\_A}(compl)} (I_{\mathbf{a}} \vee O_{\mathbf{a}})$
      $icnf := icnf \wedge \bigvee_{\mathbf{a} \in \mathsf{U\_A}(compl)} \neg U_{\mathbf{a}}$
    end if
  until $(compl \neq \varepsilon)$
  if $sstcand \neq \varnothing$ **then**
    $\mathcal{E}_{\mathsf{SST}}(\Gamma) := \mathcal{E}_{\mathsf{SST}}(\Gamma) \cup \{\mathsf{I\_A}(sstcand)\}$
    $sU := \bigwedge_{\mathbf{a} \in \mathsf{U\_A}(sstcand)} U_{\mathbf{a}}$
    $sIO := \bigwedge_{\mathbf{a} \notin \mathsf{U\_A}(sstcand)} (I_{\mathbf{a}} \vee O_{\mathbf{a}})$
    for each $sst \in \mathsf{ALLSS}(ocnf \wedge sU \wedge sIO)$ **do**
      $\mathcal{E}_{\mathsf{SST}}(\Gamma) := \mathcal{E}_{\mathsf{SST}}(\Gamma) \cup \{\mathsf{I\_A}(sst)\}$
    end for
    $ocnf := ocnf \wedge \bigvee_{\mathbf{a} \in \mathsf{U\_A}(sstcand)} \neg U_{\mathbf{a}}$
  end if
until $(sstcand \neq \varnothing)$
if $\mathcal{E}_{\mathsf{SST}}(\Gamma) = \varnothing$ **then**
  $\mathcal{E}_{\mathsf{SST}}(\Gamma) = \{\varnothing\}$
end if
return $\mathcal{E}_{\mathsf{SST}}(\Gamma)$

scale-free power-law distribution. This is generally the case when: (i) networks expand

continuously by the addition of new nodes, and (ii) new nodes attach preferentially to sites that are already well connected. Moreover, Watts and Strogatz [32] show that many biological, technological and social networks are neither completely regular nor completely random, but something in the between. They thus explored simple models of networks that can be tuned through this middle ground: regular networks *rewired* to introduce increasing amounts of disorder. These systems can be highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs, and they are named *small-world* networks by analogy with the small-world phenomenon.

The AFs have been generated by using `AFBenchGen2` [10], submitted as a possible generator for the ICCMA 17. It is worthy to emphasise that Watts-Strogatz and Barabasi-Albert produce undirected graphs: in this work, differently from Bistarelli *et al.* [4], each edge of the undirected graph is then associated with a direction following a probability distribution, that can be provided as input to `AFBenchGen2`. Such probability, provided as a parameter, varies between 0 and 1: if the parameter is 0, then the produced graph is acyclic; if it is 1, each attack is mutual.

The fourth set has been generated using the code provided in `Probo` [12] by the organisers of ICCMA-15 [27].[8] Finally, the SemiStableM set has been generated by adding to each AF of the StableM set and additional self-attacking argument.

In our experimental analysis we considered SSTMCS [31] and Aspartix [16]. All the considered benchmarks, and raw results, are available to download[9].

Experiments have been run on a cluster with computing nodes equipped with 2.5 Ghz Intel Core 2 Quad Processors, 4 GB of RAM and Linux operating system. A cutoff of 600 seconds was imposed to compute the extensions for each AF similarly to what chosen in ICCMA 17. For each solver we recorded the overall result: success (if it solved the considered problem), crashed, timed-out or ran out of memory. Unsuccessful runs—crashed, timed-out or out of memory—were assigned a runtime equal to the cutoff.

Performance are measured in terms of IPC score and Penalised Average Runtime. The IPC score, borrowed from the planning community and exploited in recent editions of the International Planning Competition [28],[10] is defined as follows. For a solvers $\mathcal{S}$ and an AFa, *score*$(\mathcal{S}, a)$ is defined as:

$$\text{s}core(\mathcal{S}, a) = \begin{cases} 0 & \textit{if } a \textit{ is not successfully analysed} \\ \frac{1}{1+\log_{10}(\frac{T_a(\mathcal{S})}{T_a^*})} & \textit{otherwise} \end{cases}$$

where $T_a(\mathcal{S})$ is the CPU time needed by a solver $\mathcal{S}$ to successfully analyse the AF a and T_a^* is the CPU-time needed by the best considered solver, otherwise. The total IPC score is the sum the scores achieved on each considered AF. Runtimes below 1.0 sec get by default the maximal score of 1.

The Penalised Average Runtime (PAR score) is a real number calculated by counting (i) runs that fail to solve the considered problem as ten times the cutoff time (PAR10)

[8] http://argumentationcompetition.org/2015/results.html

[9] https://helios.hud.ac.uk/scommv/storage/SemiStable2017

[10] http://www.icaps-conference.org/index.php/Main/Competitions

and (ii) runs that succeed as the actual runtime. PAR scores are commonly used in automated algorithm configuration, algorithm selection, and portfolio construction, [19] because using them allows runtime to be considered while still placing a strong emphasis on high instance set coverage.

In the following we rely on the Wilcoxon Signed-Rank Test (WSRT) in order to identify significant subsets of data [33]. The Wilcoxon Signed-Rank test is used for comparing performance in terms of PAR10 of two solvers. From this perspective, "no correlation" between the observed results indicates that it is equally like that, given an AF from the considered set of benchmarks, one solver provides a solution faster than the second solver, than the vice-versa. For the purposes of this analysis, the Wilcoxon sign-rank test is appropriate because it does not require any knowledge about the sample distribution, and makes no assumption about the distribution. In our analysis we considered that the null-hypothesis, i.e. the performance of compared solvers is statistically similar, is accepted when *p-value* > 0.05. Otherwise, the null-hypothesis is rejected, and therefore the compared solvers performance is statistically different.

4.2 Experimental Results

Table 1 shows the performance, in terms of PAR10, coverage and IPC score, of the considered approaches on the different testing sets.

Firstly, for each testing set, the performance of the solver that achieved the best PAR10 score are always statistically better than those of the other considered solvers.

Secondly, leaving aside the benchmarks of Barabasi—discussed in the following—AASExts shows outstanding performance. This is specially the case of Erdös-Rényi and Watts-Strogatz benchmarks, where the current state-of-the-art approaches often—if not always—fail to provide an answer in the given time. This seems consistent with some problems highlighted by Cerutti *et al.* [8] w.r.t. Aspartix in the case of preferred extensions.

The case of Barabasi-Albert benchmarks shows the main weakness of AASExts, namely the maximisation process where labels are left *free* to float between `in` and `out`. Figure 2 depicts a (small) example of an AF that would belong to the Barabasi-Albert benchmark—the actual benchmarks are composed of hundreds of arguments, Figure 2 is for illustration purpose only. Given the large occurrence of cycles in such a structure, AASExts will spend a substantial amount of time within the inner loop (cf. Algorithm 2 l. 9–16) seeking for a maximal range, especially if the first assignment from SS (cf. Algorithm 2 l. 10) contains a large set of `undec` arguments. A way to mitigate this situation is to hack the MiniSAT code in order to prioritise a specific set of variables, i.e. *injecting* in MiniSAT the knowledge that it should search towards a maximal range. It is of little surprise that SSTMCS results to be the best solver in this case since it exploits efficient techniques for computing minimal correction sets (MCS) [20, 21] that are subset-minimal sets of clauses of a formula, thus solving the dual problem of maximising the range, namely to minimise the set of `undec` arguments.

Lastly, the similarities of the results between SemiStableM and StableM suggest that the introduction of the self-defeating argument for enforcing the absence of stable extensions—cf. Example 1 and Figure 1—has no significant impact on solvers' performance. AASExts performs slightly better—according to the IPC metric—on the

Table 1. IPC score, PAR10 and coverage—percentage of AFs successfully analysed of the considered solvers—for solving the semi–stable enumeration problem on the complete testing set. Best results in **bold**.

Barabasi-Albert			
	IPC	PAR10	Coverage
Aspartix	1.1	5954.1	0.8
SSTMCS	**416.9**	**1012.6**	**84.3**
AASExts	157.8	3718.4	47.9

Erdös-Rényi			
	IPC	PAR10	Cov.
Aspartix	0.0	6000.0	0.0
SSTMCS	0.0	6000.0	0.0
AASExts	**263.0**	**2918.8**	**52.6**

SemiStableM			
	IPC	PAR10	Cov.
Aspartix	126.1	3273.0	48.8
SSTMCS	253.8	2568.1	58.2
AASExts	**312.8**	**2141.0**	**65.2**

StableM			
	IPC	PAR10	Cov.
Aspartix	116.3	3428.8	46.0
SSTMCS	242.6	2616.7	57.4
AASExts	**314.9**	**2147.8**	**65.0**

Watts-Strogatz			
	IPC	PAR10	Cov.
Aspartix	17.9	5429.0	11.0
SSTMCS	8.3	5789.9	4.6
AASExts	**395.0**	**1376.9**	**79.0**

StableM domain no doubt because a large of benchmark instances (61%) have a stable extension, and thus it can fully exploit the All-SAT solver.

However, it is interesting to note that the coverage is slightly higher in the case of SemiStableM. If an AF in StableM has a stable extension, AASExts will compute the semi-stable extensions by using STExts. However, the AF in SemiStableM, derived from the previous one by adding a self-defeating argument, will not have a stable extension and thus AASExts cannot exploits STExts. In 0.2% of AFs in SemiStableM, the

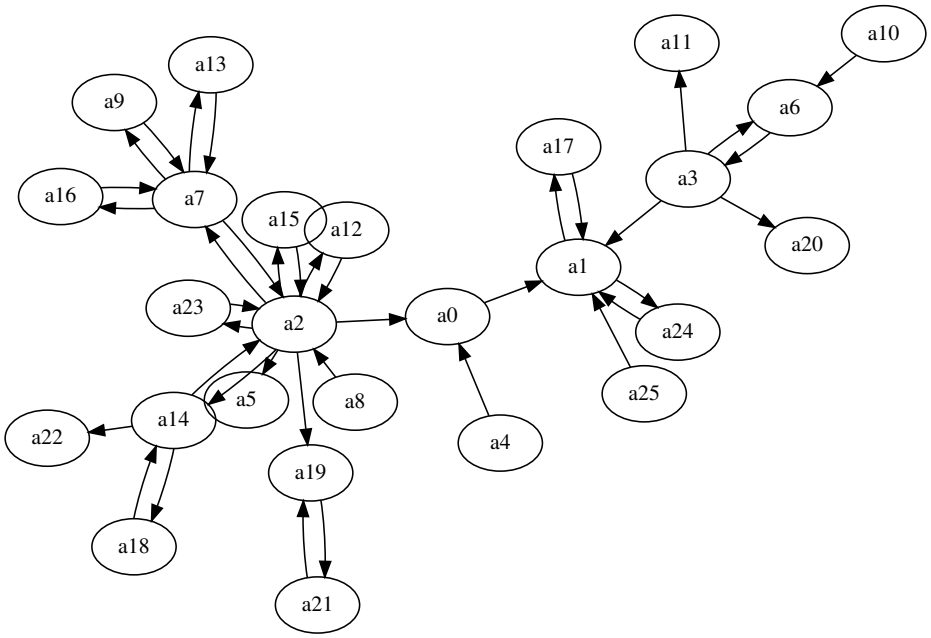

Fig. 2. A small example of an AF of the class Barabasi-Albert

procedure in AASExts identifies semi-stable extensions before the cut-off time, while it fails when searching for stable extensions in the corresponding original AF in StableM. We will investigate further this behaviour to identify the reasons those relatively rare cases are potentially problematic for the All-SAT solver exploited by STExts.

5 Conclusion

In this paper we introduced AASExts, an efficient algorithm for computing admissible argumentation stage extensions, a.k.a. semi-stable extensions, in abstract argumentation. We proved its correctness and we demonstrated its performance against existing approaches in literature, and overall this approach scored second at the ICCMA 2017 for the semi-stable semantics track. Moreover, to our knowledge, this is the first approach exploiting results from the All-SAT research to solve abstract argumentation problems.

An experimental analysis conducted on a large number of AFs based on five different graph models, has shown that: (i) AASExts is generally able to deliver better performance than existing state-of-the-art approaches; (ii) the main weakness of AASExts comes from its maximisation process, that can hardly cope with cases in which labels keep floating between `in` and `out` values, as in the Barabasi-Albert set, and (iii) the introduction of self-defeating arguments for enforcing the absence of stable extensions has no significant impact on considered solvers' performance.

As part of future work, we aim at deriving an efficient algorithm for computing (non-admissible) argumentation stage extensions, as well as for the skeptical/credulous acceptance of arguments. Moreover, we believe that it is the right time to start computing semantics evaluation considering the inner argument structure, therefore we will look at structured argumentation and how to identify argumentation stages, as well as other Dung's related semantics, possibly without the need of first deriving a Dung's argumentation framework as an intermediate system of representation.

References

1. Barabasi, A., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 11 (1999)
2. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowledge Engineering Review **26**(4), 365–410 (2011)
3. Baroni, P., Cerutti, F., Dunne, P.E., Giacomin, M.: Automata for infinite argumentation structures. Artificial Intelligence **203**(0), 104 – 150 (2013)
4. Bistarelli, S., Rossi, F., Santini, F.: Benchmarking Hard Problems in Random Abstract AFs: The Stable Semantics. In: Proceedings of COMMA. pp. 153–160 (2014)
5. Caminada, M.: On the issue of reinstatement in argumentation. In: Proceedings of JELIA 2006. pp. 111–123 (2006)
6. Caminada, M.: Semi-stable semantics. In: Proceedings of COMMA 2006. pp. 121–130 (2006)
7. Caminada, M., Gabbay, D.M.: A logical account of formal argumentation. Studia Logica (Special issue: new ideas in argumentation theory) **93**(2–3), 109–145 (2009)
8. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: A SAT-based Approach for Computing Extensions in Abstract Argumentation. In: Second International Workshop on Theory and Applications of Formal Argumentation (TAFA-13) (2013)
9. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: Computing Preferred Extensions in Abstract Argumentation: a SAT-based Approach. Tech. rep. (2013), `http://arxiv.org/abs/1310.4986`
10. Cerutti, F., Giacomin, M., Vallati, M.: Generating structured argumentation frameworks: Afbenchgen2. In: Proceedings of COMMA. pp. 467–468 (2016)
11. Cerutti, F., Giacomin, M., Vallati, M.: How we designed winning algorithms for abstract argumentation and which insight we attained. Artificial Intelligence **276**, 1 – 40 (2019). https://doi.org/https://doi.org/10.1016/j.artint.2019.08.001
12. Cerutti, F., Oren, N., Strass, H., Thimm, M., Vallati, M.: A benchmark framework for a computational argumentation competition. In: Proceedings of the 5th International Conference on Computational Models of Argument. pp. 459–460 (2014)
13. Dung, P.M.: 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)
14. Dvořák, W., Järvisalo, M., Wallner, J.P., Woltran, S.: Complexity-sensitive decision procedures for abstract argumentation. Artificial Intelligence **206**, 53–78 (2014)
15. Eén, N., Sörensson, N.: An extensible sat-solver. In: International conference on theory and applications of satisfiability testing. pp. 502–518. Springer (2003)
16. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation **1**(2), 147–177 (2010)
17. Erdös, P., Rényi, A.: On random graphs. I. Publ. Math. Debrecen **6**, 290–297 (1959)

18. Gaggl, S.A., Linsbichler, T., Maratea, M., Woltran, S.: Design and results of the second international competition on computational models of argumentation. Artificial Intelligence **279**, 103193 (2020). https://doi.org/https://doi.org/10.1016/j.artint.2019.103193
19. Hoos, H.H.: Automated algorithm configuration and parameter tuning. In: Autonomous search, pp. 37–71. Springer (2012)
20. Liffiton, M.H., Sakallah, K.A.: Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. Journal of Automated Reasoning **40**(1), 1–33 (sep 2007)
21. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. pp. 615–622. AAAI Press (2013)
22. McMillan, K.: Applying SAT Methods in Unbounded Symbolic Model Checking. In: Proc. CAV. pp. 303–323. CAV '02, Springer-Verlag, London, UK, UK (2002)
23. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**, 241–273 (1999)
24. Pu, F., Ya, H., Luo, G.: argmat-sat: Applying SAT Solvers for Argumentation Problems based on Boolean Matrix Algebra. `http://argumentationcompetition.org/2017/argmat-sat.pdf` (2017)
25. Rescher, N.: Dialectics: A controversy-oriented approach to the theory of knowledge. Suny Press (1977)
26. Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., Vallati, M.: Summary Report of The First International Competition on Computational Models of Argumentation. AI Magazine (2016)
27. Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., Vallati, M.: Summary report of the first international competition on computational models of argumentation. AI Magazine **37**(1), 102 (2016)
28. Vallati, M., Chrpa, L., Grzes, M., McCluskey, T.L., Roberts, M., Sanner, S.: The 2014 international planning competition: Progress and trends. AI Magazine **36**(3), 90–98 (2015)
29. Verheij, B.: The influence of defeated arguments in defeasible argumentation. In: WOCFAI. vol. 95, pp. 429–440. Citeseer (1995)
30. Verheij, B.: Two approaches to dialectical argumentation:admissible sets and argumentation stages. In: Meyer, J.J., van der Gaag, L.C. (eds.) Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96). pp. 357–368. Utrecht, NL (1996)
31. Wallner, J.P., Weissenbacher, G., Woltran, S.: Advanced sat techniques for abstract argumentation. In: Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems. pp. 138–154 (2013)
32. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
33. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945)
34. Yu, Y., Subramanyan, P., Tsiskaridze, N., Malik, S.: All-SAT Using Minimal Blocking Clauses. In: 2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems. pp. 86–91. IEEE (jan 2014). https://doi.org/10.1109/VLSID.2014.22