On the Impact of SAT Solvers on Argumentation Solvers

Serigne GNING [a] Jean-Guy MAILLY [b]
[a] *Université de Paris*
gning00@gmail.com
[b] *LIPADE, Université de Paris*
jean-guy.mailly@u-paris.fr

Abstract. SAT-based approaches for reasoning with abstract argumentation frameworks (AFs) have been dominating in recent years. Different SAT solvers have been used in the argumentation solvers that performed well at the different occurrences of the International Competition on Computational Models of Argumentation (ICCMA). Based on this observation, the question of the impact of the underlying SAT solver on the efficiency of the argumentation solver has arisen. We have conducted a preliminary study that shows varied results when the SAT solvers Minisat and Glucose are used. Our results suggest interesting research tracks, like the study of the link between the AF instances and the SAT solving algorithms, or the use of a portfolio of SAT solvers for reasoning with AFs.

Keywords. Abstract argumentation, SAT solvers

1. Introduction

Abstract argumentation frameworks [1] are one of the most prominent models in the domain of computational argumentation. The development of efficient computational approaches has blown up in recent years, especially in relation with the organisation of the International Competition on Computational Models of Argumentation (ICCMA) since 2015 [2,3,4]. One of the general results of the successive editions of ICCMA is the global domination of solvers based on SAT:

- CoQuiAAS [5], ArgSemSAT [6], LabSATSolver[1] and Cegartix [7] in 2015;
- pyglaf [8], ArgSemSAT, argmat-sat,[2] CoQuiAAS and argmat-dvisat [3] in 2017;
- μ-toksia [9], CoQuiAAS and pyglaf in 2019.

Moreover, SAT-based approaches have also been used for other purposes, like extension enforcement [10], status enforcement [11], or reasoning with incomplete

[1] `https://github.com/fbrns/LabSATSolver`
[2] `https://sites.google.com/site/argumatrix/argmat-sat`
[3] `https://sites.google.com/site/argumatrix/argmat-dvisat`

argumentation frameworks [12] and control argumentation frameworks [13,14]. The question that arises now is "does the choice of the SAT solver have an impact on the efficiency of the argumentation solver?". To answer this question, we have conducted a preliminary study where we compare the original CoQuiAAS solver with a modified version based on Glucose [15] instead of Minisat [16]. This choice comes from the fact that CoQuiAAS (using Minisat) was the second best solver at the last ICCMA competition, while the winner μ-toksia was based on Glucose. We have compared the efficiency of both versions of CoQuiAAS on the four classical reasoning tasks for the stable and complete semantics: given an AF, produce one extension or enumerate all the extensions; given an AF and an argument, check credulous acceptance and skeptical acceptance for this argument. Depending on the pair (task, semantics), we observe various results, that actually yield new open questions. While, in some cases, both versions of the solver are globally equivalent for most instances, in some other cases the relative efficiency of the solvers highly depends on the instance. There are also some instances that are not solved by one solver within the specified timeout, while they are quickly solved by the other solver. This suggests that there may be a link between the properties of the instances (*e.g.* the type of graph structure) and the best suited SAT solver. Thus, an interesting approach for solving efficiently different instances of AF-based reasoning may rely on a portfolio of SAT solvers.

In Section 2, we recall the basic notions of abstract argumentation, and the translations of AF semantics into propositional logic that are used by the solvers. Then, Section 3 describes our experimental protocol and results. Finally, Section 4 concludes the paper by highlighting some interesting research tracks that emerge from our experiments.

2. Background Notions

2.1. Abstract Argumentation

An argumentation framework (AF) [1] is a directed graph $F = \langle A, R\rangle$ where A is the set of arguments, and $R \subseteq A \times A$ is the attack relation. For $a, b \in A$, we say that a *attacks* b if $(a, b) \in R$. Moreover, a set $S \subseteq A$ *defends* an argument $c \in A$ if, $\forall b \in A$ s.t. $(b, c) \in R$, $\exists a \in S$ s.t. $(a, b) \in R$. Different notions of collective acceptance of arguments are defined by Dung, based on the notion of *extension*, *i.e.* some set of arguments that are jointly acceptable. An extension semantics is a function σ that maps an AF to its set of extensions. Most semantics rely on basic notions: $S \subseteq A$ is *conflict-free* iff $\forall a, b \in S$, $(a, b) \notin R$; and $S \subseteq A$ is *admissible* iff S is conflict-free and S defends all its elements.

We introduce the extension semantics that are used in this work.

- $S \subseteq A$ is a *stable extension* (denoted $S \in ST(F)$) iff S is conflict-free and $\forall b \in A \setminus S$, $\exists a \in S$ s.t. $(a, b) \in R$;
- $S \subseteq A$ is a *complete extension* (denoted $S \in CO(F)$) iff S is admissible and $\forall a \in A$ that is defended by S, $a \in S$.

We are interested in four classical reasoning problems.

SE-σ Given $F = \langle A, R\rangle$, give some $S \in \sigma(F)$.
EE-σ Given $F = \langle A, R\rangle$, give each $S \in \sigma(F)$.
DC-σ Given $F = \langle A, R\rangle$ and $a \in A$, is there $S \in \sigma(F)$ s.t. $a \in S$?
DS-σ Given $F = \langle A, R\rangle$ and $a \in A$, is $a \in S$ true for each $S \in \sigma(F)$?

We recall that reasoning with AFs is generally intractable, in particular, DC-ST and DC-CO are NP-complete, and DS-ST is coNP-complete [17]. This explains the popularity of SAT-based approaches for reasoning with AFs.

2.2. Propositional Encoding

Now, we give the propositional encodings of the stable and complete semantics that are used in SAT-based argumentation solvers, especially in CoQuiAAS [5]. These encodings come from [18]. The idea is to use argument names as propositional variables in formulas such that a set of arguments is an extension iff it is a model of the formula. More precisely, given $F = \langle A, R\rangle$ an AF, we define

- $\phi_{ST} = \bigwedge_{a\in A}(a \leftrightarrow \bigwedge_{(b,a)\in R} \neg b)$
- $\phi_{CO} = \bigwedge_{a\in A}((a \rightarrow \bigwedge_{(b,a)\in R} \neg b) \wedge (a \leftrightarrow \bigwedge_{(b,a)\in R}(\bigvee_{(c,b)\in R} c)))$

where the models of ϕ_σ exactly correspond to the σ-extensions of F, for $\sigma \in \{ST, CO\}$. Then, solving the problem SE-σ (resp. EE-σ) corresponds to computing one model (resp. all the models) of ϕ_σ, while DC-σ (resp. DS-σ) is solved by checking whether $\phi_\sigma \wedge a$ is satisfiable (resp. $\phi_\sigma \wedge \neg a$ is unsatisfiable), where a is the queried argument for acceptance.

3. Experiments

We have used two versions of CoQuiAAS: the original version, where the underlying SAT solver is Minisat, and a modified version where Minisat is replaced by Glucose. For comparing their relative efficiency, we have used the benchmark from ICCMA 2017, since they were specifically chosen for the specific semantics and reasoning tasks at hand [3]. The timeout was set to 600 seconds, and the experiments were run on Ubuntu 18.04, with a 2.16GHz CPU and 4GB of RAM.

We have first computed the Penalized Average Runtime (PAR10), *i.e.* for each reasoning task, the solvers are associated with the average runtime for all the instances, where a timeout is penalized as ten times the actual timeout. As shown on Table 1, except for SE-ST, there is no significant difference between both solvers. For all the other pairs (task, semantics), either both versions of the solvers are very efficient, or both face a high number of timeout (thus, the PAR10 score is around several thousands).

	Stable Semantics				Complete Semantics			
	EE	SE	DS	DC	EE	SE	DS	DC
CoQuiAAS + Minisat	3875.6	1.19	6.38	1.36	4854.52	375.06	1.46	1.52
CoQuiAAS + Glucose	3916.46	2175	6.39	0.48	4892.19	375.82	1.46	0.63

Table 1. PAR10 score for both versions of CoQuiAAS, rounded to 10^{-2} seconds

But then, we look at the results in more detail. We have compared the results of both solvers for each instance. Figures 1 and 2 present parts of the results, for the stable and complete semantics respectively. In each subfigure, one point corresponds to one instance, where its abscissa is the runtime obtained by CoQuiAAS with Minisat, and the ordinate is the runtime obtained by CoQuiAAS with Glucose. Some pairs (task, semantics) are omitted for space reason.

We start with the stable semantics. For DS-ST (Figure 1a), we observe that some instances are close to an axis, which means that the runtime is almost 0 for one of the solvers, but around several dozens (or even hundreds) of seconds for the other one. This is even more significant for DC-ST (Figure 1b), where almost all the instances are in this situation. This means that, some instances are hard to solve with Minisat, and easy to solve with Glucose, while some other instances are in the opposite situation. We observe again the same result for the complete semantics, see *e.g.* DC-CO at Figure 2b. A particularly interesting phenomenon concerns EE-CO (Figure 2a): many instances are solved by Minisat while reaching timeout with Glucose, and vice-versa.

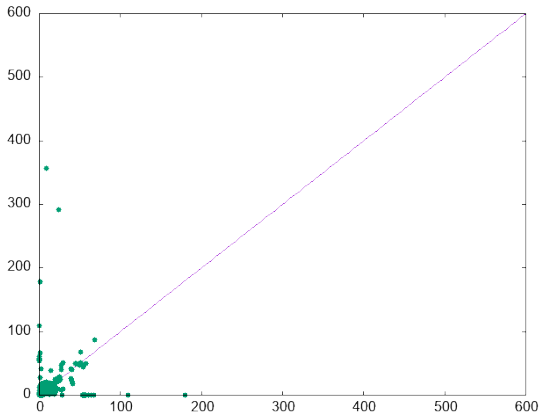
4. Conclusion

This paper presents a preliminary study of the impact of the underlying SAT solver on the efficiency of argumentation solvers. Although we wish to extend this work, with the inclusion of other SAT solvers, the existing results already suggest some interesting research tracks. Indeed, we have observed that, depending on the use of Minisat or Glucose (*i.e.* the SAT solvers that were the most successful at ICCMA 2019), some instances may be really hard to solve (even sometimes impossible to solve within the fixed time limit), or on the contrary very quick. None of the solvers has really dominated the other one. A similar study has been conducted recently [19] and draws the same conclusion. Future work includes a more detailled study of the differences between the instances that are easy to solve for one solver, and hard for the other one. Properties like *e.g.* the presence of cycles or self-attacks, or the density of attacks, may play a role in the choice of the SAT solver. Another interesting idea is to use a portfolio of SAT solvers [20]: instead of using a specific SAT solver, we can run in parallel several ones, and stop as soon as one of them finds the solution. This would allow to benefit from the power of the different SAT solving algorithms at once. Another option to optimize the use of SAT solvers consists in fine-tuning the solvers. This requires a deep analysis of how the solver's performance is affected by the type of AF for the different configurations of solvers parameters. Finally, different SAT encodings of the AF and semantics may have some effects on the SAT solvers runtime.

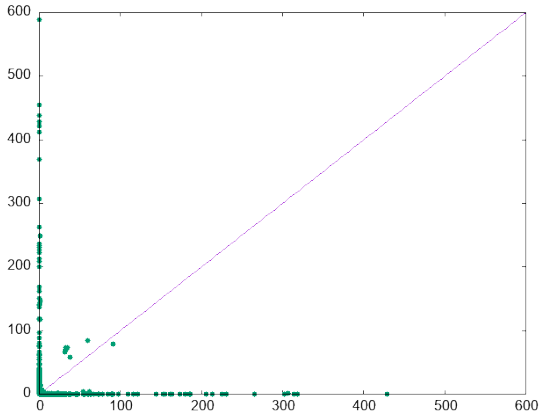
References

[1] Dung PM. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif Intell. 1995;77(2):321–358.

[2] Thimm M, Villata S. The first international competition on computational models of argumentation: Results and analysis. Artif Intell. 2017;252:267–294.



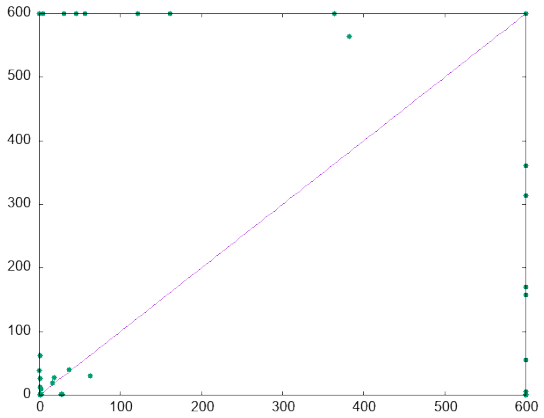
(a) DS-ST



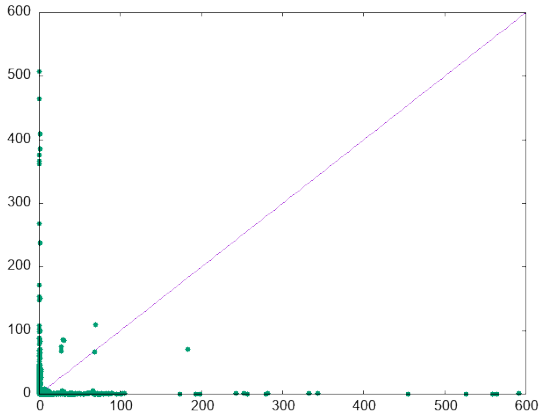
(b) DC-ST

Figure 1. Solvers runtime for the stable semantics: Minisat in abscissa; Glucose in ordinate

[3] Gaggl SA, Linsbichler T, Maratea M, Woltran S. Design and results of the Second International Competition on Computational Models of Argumentation. Artif Intell. 2020;279.
[4] Bistarelli S, Kotthoff L, Santini F, Taticchi C. Containerisation and Dynamic Frameworks in ICCMA'19. In: Proc. of SAFA 2018; 2018. p. 4–9.
[5] Lagniez J, Lonca E, Mailly JG. CoQuiAAS: A Constraint-Based Quick Abstract Argumentation Solver. In: Proc. of ICTAI 2015; 2015. p. 928–935.
[6] Cerutti F, Giacomin M, Vallati M. How we designed winning algorithms for abstract argumentation and which insight we attained. Artif Intell. 2019;276:1–40.
[7] Dvorák W, Järvisalo M, Wallner JP, Woltran S. Complexity-sensitive decision procedures for abstract argumentation. Artif Intell. 2014;206:53–78.
[8] Alviano M. Argumentation Reasoning via Circumscription with Pyglaf. Fundam Inform. 2019;167(1-2):1–30.
[9] Niskanen A, Järvisalo M. μ-toksia: An Efficient Abstract Argumentation Reasoner. In: Proc. of KR 2020; 2020. To appear.
[10] Wallner JP, Niskanen A, Järvisalo M. Complexity Results and Algorithms for Extension Enforcement in Abstract Argumentation. J Artif Intell Res. 2017;60:1–40.
[11] Niskanen A, Wallner JP, Järvisalo M. Optimal Status Enforcement in Abstract Argumentation. In: Proc. of IJCAI 2016; 2016. p. 1216–1222.



(a) EE-CO



(b) DC-CO

Figure 2. Solvers runtime for the complete semantics: Minisat in abscissa; Glucose in ordinate

[12] Niskanen A, Neugebauer D, Järvisalo M, Rothe J. Deciding Acceptance in Incomplete Argumentation Frameworks. In: Proc. of AAAI 2020; 2020. .

[13] Dimopoulos Y, Mailly J, Moraitis P. Control Argumentation Frameworks. In: Proc. of AAAI 2018; 2018. p. 4678–4685.

[14] Niskanen A, Neugebauer D, Järvisalo M. Controllability of Control Argumentation Frameworks. In: Proc. of IJCAI-PRICAI 2020; 2020. .

[15] Audemard G, Simon L. Predicting Learnt Clauses Quality in Modern SAT Solvers. In: Proc. of IJCAI 2009; 2009. p. 399–404.

[16] Eén N, Sörensson N. An Extensible SAT-solver. In: Proc. of SAT 2003; 2003. p. 502–518.

[17] Dvorák W, Dunne PE. Computational Problems in Formal Argumentation and their Complexity. In: Baroni P, Gabbay D, Giacomin M, van der Torre L, editors. Handbook of Formal Argumentation. College Publications; 2018. p. 631–688.

[18] Besnard P, Doutre S. Checking the acceptability of a set of arguments. In: Proc. of NMR 2004; 2004. p. 59–64.

[19] Klein J, Thimm M. Revisiting SAT Techniques for Abstract Argumentation. In: Proc. of COMMA 2020; 2020. To appear.

[20] Balyo T, Sanders P, Sinz C. HordeSat: A Massively Parallel Portfolio SAT Solver. In: Proc. of SAT 2015; 2015. p. 156–172.