On the Effectiveness of Automated Configuration in Abstract Argumentation Reasoning

Federico CERUTTI [a,1], Mauro VALLATI [b] and Massimiliano GIACOMIN [c]
[a] *Cardiff University, UK*
[b] *University of Huddersfield, UK*
[c] *University of Brescia, Italy*

Abstract. In this paper we investigate the impact of automated configuration techniques on the ArgSemSAT solver—runner-up of the ICCMA 2015—for solving the enumeration of preferred extensions. Moreover, we introduce a fully automated method for varying how argumentation frameworks are represented in the input file, and evaluate how the joint configuration of frameworks and ArgSemSAT parameters can have a remarkable impact on performance. Our findings suggest that automated configuration techniques lead to improved performances in argumentation solvers, an important message for participants to the forthcoming competition.

Keywords. Algorithm Configuration, Argumentation Framework Configuration, Abstract Argumentation

1. Introduction

Dung's theory of abstract argumentation [7] is a unifying framework able to encompass a large variety of specific formalisms in the areas of nonmonotonic reasoning, logic programming and computational argumentation. It is based on the notion of argumentation framework (*AF*), that consists of a set of arguments and an *attack* relation between them. Different *argumentation semantics* introduce in a declarative way the criteria to determine which arguments emerge as "justified" from the conflict, by identifying a number of *extensions*, i.e. sets of arguments that can "survive the conflict together" [4].

The first *International Competition on Computational Models of Argumentation* (ICCMA2015) determined the state-of-the-art of the current implementations for addressing the above problems with respect to the three aforementioned semantics (plus the complete extensions) [14]. In this paper we will focus on ArgSemSAT [6], that scored overall second during ICMMA2015—at one single Borda count point from the winner—despite an implementation bug discovered after the competition.

ArgSemSAT is a rather configurable solver: it allows to select different ways for encoding abstract argumentation problems in SAT, and it is able to exploit external SAT

[1]Corresponding Author: Federico Cerutti, Cardiff University, School of Computer Science & Informatics, CF24 3AA, Cardiff, UK; E-mail: CeruttiF@cardiff.ac.uk.

solvers. We manually tuned its parameter before submitting it to ICCMA2015; however, the question naturally arises: *is it possible to improve the chosen configuration?*

We investigated whether automatic configuration systems [12,1,18] can address such a question. In this work we exploit the sequential model-based algorithm configuration method SMAC [11], which represents the state of the art of configuration tools. SMAC uses predictive models of algorithm performance [13] to guide its search for good—according to a chosen metric—configurations.

Surprisingly, we also proved that the way *AF*s are described (for instance, the order in which arguments are listed) can have an effect on the overall performance. This is a remarkable finding that has been proved only (and very recently) in classical planning [16], and here for the second time. This is once again an important element that future organisers of competitions should be aware of and take into serious consideration.

Finally, for the first time—to our knowledge—we are in the position to prove that there is also a significant synergy between solvers' parameter configuration on the one side and knowledge representation (how *AF*s are described) on the other side, leading to increased performance.

Although due to space constraints we report our investigation w.r.t. ArgSemSAT only and the problem of enumeration of preferred extensions, those results can be generalised to other solvers and other semantics and problems.

Let us recall that an argumentation framework [7] consists of a set of arguments and a binary attack relation between them[2] and that preferred extensions are maximal admissible sets.

Definition 1. *An* argumentation framework *(AF) is a pair* $\Gamma = \langle \mathscr{A}, \mathscr{R} \rangle$ *where* $\mathscr{A}$ *is a set of arguments and* $\mathscr{R} \subseteq \mathscr{A} \times \mathscr{A}$*. We say that* $\boldsymbol{b}$ attacks $\boldsymbol{a}$*, or* $\boldsymbol{b} \rightarrow \boldsymbol{a}$*, iff* $\langle \boldsymbol{b}, \boldsymbol{a} \rangle \in \mathscr{R}$*.*

Given an AF $\Gamma = \langle \mathscr{A}, \mathscr{R} \rangle$*:*

- *a set* $S \subseteq \mathscr{A}$ *is a* conflict–free *set of* Γ *if* $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ *s.t.* $\boldsymbol{a} \rightarrow \boldsymbol{b}$*;*
- *an argument* $\boldsymbol{a} \in \mathscr{A}$ *is* acceptable *with respect to a set* $S \subseteq \mathscr{A}$ *of* Γ *if* $\forall \boldsymbol{b} \in \mathscr{A}$ *s.t.* $\boldsymbol{b} \rightarrow \boldsymbol{a}$*,* $\exists\, \boldsymbol{c} \in S$ *s.t.* $\boldsymbol{c} \rightarrow \boldsymbol{b}$*;*
- *a set* $S \subseteq \mathscr{A}$ *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 of* Γ*;*
- *a set* $S \subseteq \mathscr{A}$ *is a* preferred extension *of* Γ *iff S is a maximal (w.r.t. set inclusion) admissible set of* Γ*.*

2. Automated Configuration

The description of an abstract argumentation framework can be synthesised by listing all the arguments and all the attacks of the framework. Currently, three main formats for describing frameworks are used: Trivial Graph Format, Aspartix Format and the CNF Format. Here we focus on the most used one, the Aspartix Format [8].

Since this *configuration* of the input file should be performed *online* to lead to improvements of the overall system, we are interested only in information about the *AF* that can be quickly obtained. In particular, we considered the possibility to list arguments ordered according to the following five criteria: (1) the number of attacks received; (2) the

[2] In this paper we consider only *finite* sets of arguments: see [5] for a discussion on infinite sets of arguments.

number of attacks to other arguments; (3) the presence of self-attacks; (4) the difference between the number of received attacks and the number of attacks to other arguments; and (5) being an argument in a mutual attack. For each of the five mentioned criteria, arguments can be listed following a direct or inverse order.

To order the list of attacks, these five criteria can be applied either to the attacking or to the attacked argument. The choice of the criteria for ordering the list of arguments is independent from the choice of criteria for ordering the list of attacks.

There are different ways for encoding the degrees of freedom in *AF*s descriptions as parameters, mainly because orders are not natively supported by general configuration techniques. Following [16], we generate 10 continuous parameters, which correspond to the aforementioned possible orderings of arguments and attacks in frameworks. An additional categorical selector among 5 alternatives allows to decide how to apply the criteria for ordering the list of attacks, i.e. whether on the first or the second argument, and following same or inverse ordering of arguments.

Each continuous parameter has associated a real value in the interval $[-1.0, +1.0]$ which represents (in absolute value) the *weight* or *precedence* given to an ordering criterion: the criterion corresponding to the parameter with the highest absolute value, is considered first in the ordering. Ties of such ordering are then broken by referring to the criterion associated to the next parameter of high absolute value. Negative values indicate that inverse ordering is used. In the case of two criteria having exactly the same absolute value, they are applied according to their alphabetical ordering. Thus, the configuration space is $\mathscr{C} = [-1.0, +1.0]^{10} \cdot 5$, where 5 are the possible values of the categorical parameter describing the order of the list of attacks.

In order to automatically re-order an argumentation framework according to the specified configuration, we developed a wrapper in Python. On the *AF*s considered in our experimental analysis, composed by hundreds of arguments and few hundreds of thousands of attacks, the re-ordering of the Aspartix format description takes less than 1 CPU-time second.

Joint AF-Solver Configuration As a case study for investigating the synergies of re-ordering a given argumentation framework, and of selecting the most appropriate solver's parameters, we consider ArgSemSAT [6], which is the runner-up of ICCMA 2015. On the one hand, ArgSemSAT exposes a single—critical—parameter which allows to select the encoding for translating the problem of identifying a complete extension into a SAT formula, with remarkable impact on size and structure of the generated CNFs, and on the CPU-time required to enumerate all the preferred extensions. On the other hand, ArgSemSAT allows the use of an external SAT solver, to be used as an NP-oracle. In this work we exploit the Glucose SAT solver [2]: it shows very good performance in recent SAT competitions, and has a large number of parameters that can be tuned and controlled for modifying its behaviour, from decay level of variables and clauses, to the number of restarts. Configuring ArgSemSAT together with Glucose requires to tune 20 parameters (2 categorical and 18 continuous).

In order to maximise the impact of automated configuration on solvers' performance and thus exploiting unforeseen synergies between solver behaviour and specific knowledge descriptions, we use SMAC for configuring at the same time the *AF*s description

and the configuration of ArgSemSAT. The total number of configurable parameters is 31: 3 categorical and 28 continuous.[3]

SMAC [11] is an *anytime algorithm* (or *interruptible algorithm*) that interleaves the exploration of new configurations with additional runs of the current best configuration to yield both better and more confident results over time. As all anytime algorithms, SMAC improves performance over time, and for finite configuration spaces it is guaranteed to converge to the optimal configuration in the limit of infinite time.

3. Experimental Analysis

Settings. As described in the previous section, in this work we consider ArgSemSAT using Glucose as SAT solver [2] for enumerating preferred extensions. In total, 31 parameters are exposed. Three of them are categorical, while the others are continuous.

We randomly generated 8,000 *AF*s, divided into 4 sets of 2,000 *AF*s each. Three of such sets include only framework based on different graph models: Barabasi-Albert [3], Erdös-Rényi [9] and Watts-Strogatz [17]. The fourth set ("General") includes mixed-structured *AF*s generated by considering graphs of all the mentioned models.

To identify challenging frameworks *AF*s we followed the protocol suggested in [15] which leads to the selection of *AF*s with a number of arguments between 250 and 650, and number of attacks between (approximately) 400 and 180,000.

Each set of *AF*s has been split into a training set (1,800 *AF*s) and a testing set (200 *AF*s) in order to obtain an unbiased estimate of generalisation performance to previously unseen *AF*s from the same distribution.

Configuration was done using SMAC version 2.10. The performance metric we optimised is the Penalized Average Runtime (PAR), counting runs that crash or do not find a solution as ten times the cutoff time (PAR10).

Experiments were performed on Dual Xeon X5660-2.80GHz with 48GB DDR3 RAM. Each configuration run was limited to a single core, and was given an overall runtime of 5 days and 4 GB of RAM, for ensuring re-usability of results also on less equipped machines. The cutoff time was 500 seconds.

In the following, also the IPC score is used for comparing different configurations performance. For a solver $\mathscr{C}$ and a problem p, $Score(\mathscr{C}, p)$ is 0 if p is unsolved, and $1/(1+\log_{10}(T_p(\mathscr{C})/T_p^*))$ otherwise (where T_p^* is the minimum amount of time required by any compared system to solve the enumeration problem). The IPC score on a set of instances is given by the sum of the scores achieved on each considered instance.

Results. Table 1 compares the performance of ArgSemSAT using the default configuration, and the specific joint configuration of *AF*s description and ArgSemSAT, obtained by running SMAC. Remarkably, the joint configuration of *AF* description and ArgSemSAT leads to a general performance improvement. In particular, on the Barabasi-Albert and General sets the performance of the configured system are statistically significantly better than the performance achieved by using the default configuration, according to the Wilcoxon test. The significant performance improvement achieved on the General set is of particular interest: it indicates that it is possible to identify a configuration able to

[3]The interested reader can find the full list of parameters, including default value and valid value range, at `https://helios.hud.ac.uk/scommv/afconf/PARAMS.TXT`.

Table 1. Comparison between the default and tuned configuration, in terms of IPC score, PAR10, and percentage of instances on which a configuration has been the fastest, on the considered *AF*s test sets for enumerating preferred extensions. In bold the best results.

Set	Configuration	IPC Score	PAR10	Fastest
Barabasi-Albert	Default	78.0	1921.0	2.5
	Configured	**125.2**	**1863.1**	**60.5**
Erdös-Rényi	Default	56.8	3426.5	16.5
	Configured	**60.4**	**3329.2**	**18.0**
Watts-Strogatz	Default	116.6	**1967.3**	**28.0**
	Configured	**118.1**	1967.9	23.5
General	Default	110.0	1665.4	11.0
	Configured	**143.0**	**1376.8**	**62.5**

improve the performance across differently-structured graphs. In other words, this is an indication that the default configuration can be improved.

Conversely, the configuration process does not significantly improve the default performance on the Watts-Strogatz set. According to the Wilcoxon test, performance of default and tuned configurations are statistically undistinguishable even though IPC score show slight improvements. This is possibly due to the fact that the default configuration is already showing very good performance. In that scenario, it may be the case that only small portions of the configuration space lead to a significant performance improvement over the default configuration. Given the limited CPU-time made available to the configuration process, SMAC did not identify such portions of the vast configuration space.

Finally, the results on the Erdös-Rényi set deserves a more detailed discussion. On the one hand, the Wilcoxon test indicates that there is not a statistically significant performance improvement. On the other hand, *AF*s from the Erdös-Rényi set are extremely hard for ArgSemSAT, as testified by the PAR10 values. Moreover, those that can be solved are usually solved quickly, i.e. in few CPU-time seconds. This makes the evaluation of configurations' performance, and the exploration of the space of configurations, hard and slow. Despite such issues, SMAC was able to identify a configuration that is able to improve the performance both in terms of runtime (better IPC score) and PAR10. Overall, this is a remarkable result, that shows the ability of automated configuration in improving performance also in unfavourable cases.

To provide a better overview of the impact of different configurations, we ran all the configurations obtained by SMAC from training sets with different graph models on all the considered test sets. Table 2 shows the results of this comparison. Performance of different configurations tend to be similar but for the parameters' configuration derived from Barabasi-Albert training *AF*s. This possibly indicates that there are some parameters' values that can boost performance on differently-structured *AF*s. Remarkably, the configuration identified by training on the General set, is usually able to obtain performance that are close to those of the specific configuration, on each considered test set. Again, this supports the hypothesis that there are some parameters' values that can help improving the general performance. On the contrary, Table 2 also indicates that the configuration derived from Barabasi-Albert training does not generalise well on differently-structured *AF*s. Such behaviour is possibly due to some parameter's value that helps increasing the performance on the Barabasi-Albert test set, but has a detrimental effect on different graph structures. For instance, among considered structures, Barabasi-Albert is

Table 2. Performance of each configuration generated by SMAC (rows) running on the different test sets (columns) for enumerating preferred extensions. IPC Score is evaluated by considering all the four configurations on each single test set. In bold the best results, with respect to each specific test set.

Training sets	**Test sets**			
	Barabasi-Albert	Erdös-Rényi	Watts-Strogatz	General
Barabasi-Albert	**119.2**	6.9	34.5	42.8
Erdös-Rényi	92.3	**58.6**	105.3	125.7
Watts-Strogatz	116.2	52.6	**115.6**	129.2
General	87.5	57.6	113.5	**133.2**

Table 3. Most important single parameters (configured value) for SMAC runs on the considered *AF* sets. F-,S- and G- stand for, respectively, Framework, ArgSemSAT and Glucose parameters.

Set	**1st**	**2nd**	**3rd**
Barabasi-Albert	S-ExtEnc (011111)	G-firstReduceDB (1528)	G-cla-decay (0.32)
Erdös-Rényi	F-autoFirst (-1.00)	G-rnd-freq (0.00)	G-K (0.26)
Watts-Strogatz	S-ExtEnc (101010)	G-Grow (0)	G-rnd-freq (0.08)
General	S-ExtEnc (101010)	G-R (2.09)	G-cla-decay (0.99)

the only set of *AF*s with a large number of preferred extensions (up to some thousands) per *AF*.

Discussion. In order to shed some light on the usefulness of algorithm and *AF* tuning, we used fANOVA [10], a recently-released tool for assessing parameter importance after each configuration. fANOVA exploits predictive models of the performance of each configuration for assessing the importance of each parameter, regardless of the value of the others, and the interaction between parameters' values. Table 3 shows the three most important parameters for each configuration. Unsurprisingly, the encoding used by ArgSemSAT for generating the SAT formulae is usually the most important parameter. Its default value (i.e. 101010), is proven to be the best choice for *AF*s belonging to Watts-Strogatz and General sets, but not for *AF*s in the Barabasi-Albert set. After that, the parameters that control the behaviour of Glucose are those with the highest impact on performance, notably: decay value of clauses and size of the DB of learnt clauses are among the aspects with a strongest impact on the performance of ArgSemSAT.

One parameter used for controlling the *AF* description has a significant impact on performance on *AF*s belonging to the Erdös-Rényi set, according to the fANOVA tool. In this case, the order in which arguments are listed is important and, in particular, it is required that self-attacking arguments are provided at the very end.

However, the interaction of parameters controlling the shape of *AF*s with reasoning-related parameters do have a remarkable impact, i.e. the best performance depends on two or more parameters. Parameters used for controlling the order of arguments have strong interactions with the parameter that controls the encoding of ArgSemSAT, as well as with parameters of Glucose controlling the number and type of clauses learnt. Figure 1 (coloured) shows the average PAR10 performance of ArgSemSAT on the Barabasi-Albert set as a function of two interacting parameters. `args_eachOther` is used for listing earlier in the *AF* description arguments that are attacking each other, the other parameter is used for controlling the number of Glucose learnt clauses, according to their

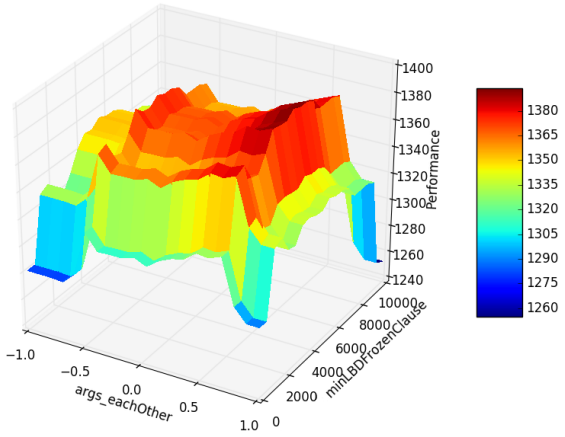

Figure 1. (Coloured) The average PAR10 performance of ArgSemSAT on the Barabasi-Albert set, as a function of the ordering of arguments according to the fact that they attack each other (`args_eachOther`) and of number of clauses stored by Glucose during the search. Lower PAR10 values correspond to better performance.

heuristic LBD evaluation. In order to achieve better performance in terms of PAR10, arguments attacking each other should be listed very late (-1.0 value of the parameter) or very early ($+1.0$ value) and either few or many clauses should be kept (respectively, low and high value of Glucose parameter).

Parameters that control the order in which arguments are listed tend to have a stronger impact on overall performance—either singularly (Table 3) or as a result of their interaction with other parameters (Fig. 1)—than parameters controlling the order in which attacks are listed. At a first sight, this may be seen as counter-intuitive, since the number of attacks in a typical benchmark *AF* is significantly higher than the number of arguments. However, this difference can be due to the data structure used by ArgSemSAT. The set of arguments of the *AF* is stored in a list which is populated according to the order in which the arguments are listed in the input file. Each argument has then an associated data structure with pointers to two other lists of arguments: one for the attacked arguments; and one for the arguments that attack it. Then the list representing the set of arguments of the *AF* is navigated several times when creating CNFs to be evaluated by the SAT solver: these results suggest that not only the encoding of complete labellings in CNF, but also the order of clauses have a remarkable impact on the performance.

4. Conclusion and Future Work

In this paper we proposed an approach for the joint automatic configuration of *AF* descriptions and argumentation solvers. Specifically, we designed a method to automatically order the list of arguments and the list of attacks in argumentation frameworks by tuning 11 parameters, using as a test-case the widely used Aspartix format. We focused our investigation on ArgSemSAT—runner-up of the ICCMA2015—using Glucose as a SAT solver: they export together a further set of 20 parameters.

As described in the previous sections: (i) we demonstrate that joint AF-solver configuration has a statistically significant impact on the performance of ArgSemSAT; (ii)

we demonstrate the synergies between *AF*s configuration and SAT solvers behaviour; and (iii) we open new, exciting possibilities in the area of learning for improving performance of abstract argumentation solvers. We believe this work would be particularly beneficial for the participants of the forthcoming competition ICCMA2017.

We see several avenues for future work. We plan to evaluate the proposed joint *AF*-solver configuration approach on different solvers and on different problems and on different semantics. Moreover, we are interested in exploiting the configuration approach for combining different argumentation and SAT solvers into portfolios. Finally, we are considering investigating the presence of *AF* configurations that are able to improve—on average—the performance of all the existing state-of-the-art argumentation solvers: this would provide powerful guidelines for the encoding of frameworks.

Acknowledgement
This work was performed using the computational facilities of the Advanced Research Computing @ Cardiff (ARCCA) Division, Cardiff University.

References

[1] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Proc. of CP*, pages 142–157, 2009.

[2] G. Audemard and L. Simon. Lazy clause exchange policy for parallel sat solvers. In *SAT 2014*, pages 197–205. 2014.

[3] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):11, 1999.

[4] P Baroni, M Caminada, and M Giacomin. An introduction to argumentation semantics. *Knowledge Engineering Review*, 26(4):365–410, 2011.

[5] P. Baroni, F. Cerutti, P. E. Dunne, and M. Giacomin. Automata for Infinite Argumentation Structures. *Artif. Intell.*, 203(0):104–150, 2013.

[6] F. Cerutti, M. Vallati, and M. Giacomin. Argsemsat-1.0: Exploiting sat solvers in abstract argumentation. *System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15)*, page 4, 2015.

[7] P. M Dung. On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. *Artif. Intell.*, 77(2):321–357, 1995.

[8] U. Egly, S. A. Gaggl, and S. Woltran. Aspartix: Implementing argumentation frameworks using answer-set programming. In *Logic Programming*, pages 734–738. 2008.

[9] P. Erdös and A. Rényi. On random graphs. I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[10] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proc. of ICML*, pages 754–762, 2014.

[11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION*, pages 507–523, 2011.

[12] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: An automatic algorithm configuration framework. *J. Artif. Intell. Res.*, 36:267–306, 2009.

[13] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artif. Intell.*, 206:79–111, 2014.

[14] M. Thimm, S. Villata, F. Cerutti, N. Oren, H. Strass, and M. Vallati. Summary report of the first international competition on computational models of argumentation. *AI Magazine*, 2016.

[15] M. Vallati, L. Chrpa, M. Grzes, T. L. McCluskey, M. Roberts, and S. Sanner. The 2014 international planning competition: Progress and trends. *AI Magazine*, 2015.

[16] M. Vallati, F. Hutter, L. Chrpa, and T. L. McCluskey. On the effective configuration of planning domain models. In *Proc. of IJCAI*, 2015.

[17] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[18] Z. Yuan, T. Stützle, and M. Birattari. Mads/f-race: Mesh adaptive direct search meets f-race. In *Proc. of IEA/AIE*, pages 41–50, 2010.