April 24, 2018

SAT-based Approaches to Adjusting, Repairing, and Computing Largest Extensions of Argumentation Frameworks

Tuomo LEHTONEN [a], Andreas NISKANEN [a] and Matti JÄRVISALO [a]
[a] *Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland*

Abstract. We present a computational study of effectiveness of declarative approaches for three optimization problems in the realm of abstract argumentation. In the *largest extension* problem, the task is to compute a σ-extension of largest cardinality (rather than, e.g., a subset-maximal extension) among the σ-extensions of a given argumentation framework (AF). The two other problems considered deal with a form of dynamics in AFs: given a subset S of arguments of an AF, the task is to compute a closest σ-extension within a distance-based setting, either by *repairing* S into a σ-extension of the AF, or by *adjusting* S to be a σ-extension containing (or not containing) a given argument. For each of the problems, we consider both iterative Boolean satisfiability (SAT) based approaches as well as directly solving the problems via Boolean optimization using maximum satisfiability (MaxSAT) solvers. We present results from an extensive empirical evaluation under several AF semantics σ using the ICCMA 2017 competition instances and several state-of-the-art solvers. The results indicate that the choice of the approach can play a significant role in the ability to solve these problems, and that a specific MaxSAT approach yields quite generally good results. Furthermore, with impact on SAT-based AF reasoning systems more generally, we demonstrate that, especially on dense AFs, taking into account the local structure of AFs can have a significant positive effect on the overall solving efficiency.

Keywords. abstract argumentation, argumentation dynamics, constraint optimization, experimentation

1. Introduction

The study of computational models of argumentation is today a vibrant area of artificial intelligence research [1] with promising applications in various fields [2]. The extension-based setting of abstract argumentation frameworks (AFs) [3] is one of the central formalisms for representing and reasoning about knowledge within argumentation. Within the last few years, a variety of systems for reasoning over AFs have emerged [4], the development of which is also incentivized today by the ICCMA series of competitions for implementations of AF reasoning systems [5].

Argumentation is intrinsically a dynamic process in which different conflicting points of views evolve through interaction. While a majority of attention on implementing reasoning over AFs has focused on central "static" problems, such as deciding whether a given argument is credulously or sceptically accepted in a given AF, the study of argumentation dynamics has recently been investigated from various perspectives, including expansions (see e.g. [6,7]), revision (e.g. [8,9,10,11,12]), and enforcement (e.g. [13,14,15,16,17]).

This work constitutes a computational study of algorithmic approaches to three NP-hard optimization problems addressing different aspects of argumentation frameworks. In terms of static problems, we consider the problem of finding a *largest extension*—in contrast to, e.g., a subset-maximal admissible set or a preferred extension, of a given AF, with the intuition of finding a most open-minded or generally accepted viewpoint. In terms of AF dynamics, we present a first computational study of dealing with the problems of *adjusting* and *repairing* a given subset of arguments in terms of a given AF [18]. In contrast to e.g. enforcing a set of arguments to constitute an extension by changing the attack relation of an AF, in adjust and repair changes are applied to an actual subset of arguments. More specifically, here the aim is to modify a set of arguments (by removing or adding arguments to it) so that the set becomes an extension of a given (possibly earlier changed) AF. The goal is to make the least adjustments possible within a distance-based setting [19], thereby from the computational perspective giving rise to combinatorial optimization problems. In other words, we focus on computational approaches to adjusting and repairing a point of view which is or has become unacceptable in terms of the current state of the world, represented as an AF, due to e.g. revising the AF by adding new arguments [6,7] or modifying the attack structure [13] to accommodate new knowledge.

Motivated by the success of applications of Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT) solvers within AF systems for static and dynamic problems (see, e.g., [20,21,22]), we consider several variants of both iterative SAT-based approaches and direct MaxSAT encodings for each of these three problems. We present results from an extensive empirical evaluation of these approaches on ICCMA 2017 instances, focusing as first choices on three AF semantics (admissible, complete, and stable). To the best of our knowledge, the approaches we study here are the first considered for the dynamic problems of adjust and repair. In terms of computing a largest extension, we expand on previous work [23] by considering further alternative approaches. The results suggest guidelines for approaches to consider for developing computational techniques for solving optimization problems arising from AF dynamics (among others). Additionally, applicable to standard SAT encodings of AF semantics in general and thereby of interest more generally for developing SAT-based argumentation systems, we show that by taking into account the local structure of the AF at hand, the SAT encodings can be generated faster, made smaller, and also can become noticeably faster to solve compared to exactly implementing the clause generation according to the formal definition of the SAT encodings.

2. Argumentation Frameworks

We shortly recall argumentation frameworks [3] and their semantics [24].

Definition 1. An *argumentation framework (AF)* is a pair $F = (A, R)$, where A is a finite set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a,b) \in R$ means that a attacks b. An argument $a \in A$ is *defended* (in F) by a set $S \subseteq A$ if, for each $b \in A$ such that $(b,a) \in R$, there exists a $c \in S$ such that $(c,b) \in R$.

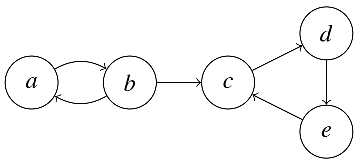

Figure 1. Argumentation framework from Example 1.

Example 1. Let $F = (A,R)$ be an AF with the set of arguments $A = \{a,b,c,d,e\}$ and the attack relation $R = \{(a,b),(b,a),(b,c),(c,d),(d,e),(e,c)\}$. The AF F is represented as a directed graph in Figure 1.

Semantics for argumentation frameworks are functions σ assigning to each AF F a collection $\sigma(F) \subseteq 2^A$ of σ-extensions (i.e., sets of jointly acceptable arguments with respect to σ). In this work, we consider the admissible, complete, and stable semantics. For the definitions we require the concepts of the characteristic function and the range.

Definition 2. For an AF $F = (A,R)$, the *characteristic function* $\mathscr{F}_F : 2^A \to 2^A$ is $\mathscr{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$. The *range of* $S \subseteq A$ is $S_R^+ = S \cup \{x \mid (y,x) \in R, y \in S\}$.

The semantics considered here are then defined as follows.

Definition 3. 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 $(a,b) \in R$. We denote the collection of conflict-free sets of F by $cf(F)$. For a conflict-free set $S \in cf(F)$ it holds that (i) $S \in stb(F)$ (S is a stable extension) iff $S_R^+ = A$, (ii) $S \in adm(F)$ (S is admissible) iff $S \subseteq \mathscr{F}_F(S)$, and (iii) $S \in com(F)$ (S is complete) iff $S = \mathscr{F}_F(S)$.

It is well-known that for any AFs F we have $stb(F) \subseteq com(F) \subseteq adm(F)$.

Example 2. For the AF F in Example 1 we have $stb(F) = \{\{b,d\}\}$, $adm(F) = \{\emptyset, \{a\}, \{b\}, \{b,d\}\}$, and $com(F) = \{\emptyset, \{a\}, \{b,d\}\}$.

3. Computational Problems

We continue by formally defining the computational problems considered in this work. For the following, we standardly denote the cardinality of a finite set S by $|S|$, and define $\arg\max_{x \in S} f(x) = \{x \in S \mid \forall y \in S\colon f(y) \leq f(x)\}$ and $\arg\min_{x \in S} f(x) = \{x \in S \mid \forall y \in S\colon f(y) \geq f(x)\}$.

The maximization problem of Largest Extension asks, given an AF F and a semantics σ, to output a largest (in terms of set-cardinality) σ-extension in the AF F.

LARGEST EXTENSION
Input: AF $F = (A,R)$ and semantics σ.
Task: Find an E^* such that
$$E^* \in \arg\max_{E \in \sigma(F)} |E|.$$

Note that the solutions to Largest Extension coincide for $\sigma \in \{adm, com\}$.

We also consider the optimization problems of repairing and adjusting an AF [18], as forms of AF dynamics. The corresponding decision problems are W[1]-hard in general when parameterized by the Hamming distance [18], and therefore the optimization problems we consider here are not expected to be solvable in polynomial time.

For the following, the Hamming distance between two finite sets S_1 and S_2 is the cardinality of the symmetric difference $|S_1 \Delta S_2| = |S_1 \setminus S_2| + |S_2 \setminus S_1|$.

The minimization problem Repair asks, given an AF F, a subset S of arguments, and a semantics σ, to find a non-empty σ-extension closest to S with respect to the Hamming distance. In other words, we are asked to modify the set S into a σ-extension with the minimum amount of additions and deletions of arguments.

REPAIR
Input: AF $F = (A, R)$, $S \subseteq A$, and semantics σ.
Task: Find an E^* such that

$$E^* \in \operatorname*{arg\,min}_{E \in \sigma(F):\ E \neq \emptyset} |E \Delta S|.$$

Finally, an instance of the problem Adjust consists of an AF F, a semantics σ, a σ-extension E_0 of the AF, and an argument $t \in A$. The task is to find another σ-extension closest to E_0 and, if $t \notin E_0$ (resp. $t \in E_0$), containing (resp. not containing) t. In other words, the task is to modify E_0 into a σ-extension (not) containing t with as few additions and deletions as possible.

ADJUST
Input: AF $F = (A, R)$, semantics σ, $E_0 \in \sigma(F)$, and $t \in A$.
Task: Find an E^* such that

$$E^* \in \operatorname*{arg\,min}_{E \in \sigma(F):\ t \in E \Delta E_0} |E \Delta E_0|.$$

Example 3. Consider again the AF F in Example 1, the extensions of which enumerated in Example 2. Fix the semantics $\sigma = adm$. Now the optimal solution to Largest Extension with F as input is clearly $\{b, d\}$. Let $S = \{a, b\}$. Given F and S as input to the Repair problem, there are now two optimal solutions—$\{a\}$ and $\{b\}$—since the Hamming distance to S is in both cases exactly one. Consider now the extension $\{b\}$ and the argument d. Now the only solution to the Adjust problem with input F, $E_0 = \{b\}$, and $t = d$ is $\{b, d\}$, with Hamming distance one.

4. SAT-Based Approaches

We now detail the computational approaches the relative efficiency of which we evaluate in this work. We focus on what we refer to SAT-based approaches, i.e., using Boolean satisfiability solvers and their extensions to Boolean optimization (MaxSAT solvers).

4.1. SAT and MaxSAT

For a variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assignment is a function from variables to $\{0, 1\}$. A clause c is satisfied by a truth assignment τ ($\tau(c) = 1$) if $\tau(x) = 1$ for a literal x in c, or $\tau(x) = 0$ for a literal $\neg x$ in c;

otherwise τ does not satisfy c ($\tau(c) = 0$). The NP-complete Boolean satisfiability problem (SAT) asks whether there is a truth assignment that satisfies a given set of clauses, i.e., a propositional formula in conjunctive normal form (CNF).

An instance $\varphi = (\varphi_h, \varphi_s)$ of the *Partial MaxSAT* problem consists of a set φ_h of *hard* clauses and a set φ_s of *soft* clauses. Any truth assignment τ that satisfies every clause in φ_h is a *solution* to φ. The *cost* of a solution τ to φ is $\textsc{cost}(\varphi, \tau) = \sum_{c \in \varphi_s}(1 - \tau(c))$, i.e., the number of soft clauses not satisfied by τ. A solution τ is *optimal* for φ if $\textsc{cost}(\varphi, \tau) \leq \textsc{cost}(\varphi, \tau')$ holds for any solution τ' to φ. Given φ, the Partial MaxSAT problem asks to find an optimal solution to φ. From here on, we refer to Partial MaxSAT simply as MaxSAT.

4.2. *Encoding AF semantics as SAT*

As a basis of the SAT and MaxSAT approaches, we employ the standard CNF encodings of the AF semantics, following [25]. Let $F = (A, R)$ be an AF. Define for each argument $a \in A$ a Boolean variable x_a. Given a semantics σ, the variables set to true in the formula $\varphi_\sigma(F)$ encode a σ-extension of F, i.e., $\{a \in A \mid \tau(x_a) = 1\} \in \sigma(F)$ for any satisfying truth assigment τ of $\varphi_\sigma(F)$. Now conflict-free sets can be expressed via the propositional formula $\varphi_{cf}(F) = \bigwedge_{(a,b) \in R}(\neg x_a \vee \neg x_b)$, stating that no attacks can occur in the extension. Admissible sets are encoded by $\varphi_{adm}(F) = \varphi_{cf}(F) \wedge \bigwedge_{(b,a)\in R}\left(x_a \rightarrow \left(\bigvee_{(c,b)\in R} x_c\right)\right)$, requiring that each argument is defended by the extension. Complete semantics have the additional requirement of including every argument that is defended by the extension: $\varphi_{com}(F) = \varphi_{adm}(F) \wedge \bigwedge_{a \in A}\left(\left(\bigwedge_{(b,a)\in R}\left(\bigvee_{(c,b)\in R} x_c\right)\right) \rightarrow x_a\right)$. Finally, stable semantics is encoded by the formula $\varphi_{stb}(F) = \varphi_{cf}(F) \wedge \bigwedge_{a\in A}\left(\neg x_a \rightarrow \bigvee_{(b,a)\in R} x_b\right)$.

4.3. *Extension Enumeration for Largest Extension*

For the Largest Extension problem, a baseline approach is to enumerate all extensions. Intuitively, this approach can be reasonable especially when the number of extensions of a given AF is small. A refinement of this approach we will consider in our experiments is to iteratively rule out all subsets of the already enumerated extensions from subsequent enumeration. For implementing this idea, we use a SAT solver incrementally: after each found solution, representing an extension E, we add the clause $\bigvee_{a \in A \setminus E} x_a$ to the solver, and continue the SAT search without altering the state of the solver otherwise.

4.4. *Iterative SAT*

Intuitively, the cost functions for each of the problems of Largest Extension, Repair, and Adjust, can be seen as computing the Hamming distance k to a subset of arguments of particular size. For solving each of the problems exactly, a SAT solver can be used in an iterative manner, iterating over the possible values $k \in \{0, \ldots, |A|\}$. For each value of k, the SAT solver call will consist of asking whether there is an extension (i) within distance $|A| - k$ from A in the Largest Extension problem, (ii) within distance k from S in Repair, and (iii) within distance k from E_0, (not) containing t in Adjust. In practice, the clauses for the SAT solver calls consist of the encoding of the semantics at hand and an encoding of a cardinality constraint, bounding the distance within k. For Adjust we also have the additional *hard clause* (x_t) (if $t \in A \setminus E_0$) or $(\neg x_t)$ (if $t \in E_0$), to ensure that t is properly taken into account.

For the iteration process, we will consider the following iteration strategies.

Stepping: A simple strategy is to increment $k = 0, \ldots$ linearly, and to terminate when the satisfiability status changes from the previous SAT solver calls. For Largest Extension, termination happens when the SAT solver reports unsatisfiability, at which time the extension found via the last satisfiable solver call is an optimal solution. This strategy is refined by incrementing k to the size of the latest found extension + 1 (instead of a unit increment). For Repair and Adjust, this strategy repeatedly asks for an extension within $k = 0, \ldots$ changes to the current set of arguments until a call is satisfiable, giving a closest extension.

Binary: The binary search strategy implements binary search over $k \in [0, |A|]$. For Largest Extension, this amounts to iteratively querying whether there are extensions greater than or equal to the midpoint between the current lower and upper bounds, until the bounds meet. For Repair and Adjust, the SAT solver calls constitute asking for an extension within distance equal to the midpoint from the given set of arguments.

Progressive: Another strategy is to start the search from $k = 0$, but instead of the conservative increments of Stepping, increase k in progressively larger steps: after each call, the size of the step by which k is increased is doubled, until a limit is reached. The limits are (i) surpassing the upper bound, and (ii) the call being unsatisfiable for Largest Extension and the call being satisfiable for Repair and Adjust. When either of these conditions is met, the size of the increase is returned to 1 and the search continues from the lower bound. The answer is reached when the bounds meet. A modification of this strategy for Largest Extension is to not only increase k by the value of the step, but to first increase k to be equal to the size of the found extension.

For Largest Extension, an initial lower bound on k is obtained by making the first SAT solver call without enforcing the cardinality constraint, and taking the distance obtained from the extension reported by the SAT solver as the lower bound.

For encoding the required cardinality constraints as clauses, we consider two alternative cardinality constraint encodings that are often used in practice: sequential counter [26] and cardinality networks [27]. The sequential counter encoding implements a sequential counter circuit via $\mathcal{O}(nk)$ clauses and $\mathcal{O}(nk)$ auxiliary variables for an $\leq k$ cardinality constraint over n variables, with the intuition of computing partial sums of over the inputs, sequentially considering an increasing number of the input bits. Cardinality networks implement sorting algorithms based on pair-wise comparisons of bits using $\mathcal{O}(n \log^2 n)$ clauses and auxiliary variables. At the same time, and importantly for iterative applications, they allow for incremental use of SAT solvers for iterating over k, since the cardinality constraint for different values of k can be enforced by assuming that the kth bit of the output of the network takes the value 0, i.e., without changing the structure of the actual network or its clausal encoding.

4.5. MaxSAT

Instead of using a hand-crafted iterative SAT approach, another more direct approach to solving the considered optimization problems of Largest Extension, Repair, and Adjust, is to employ a MaxSAT solver. For this approach, the clauses encoding the AF semantics are considered hard clauses. Unit soft clauses over the x_a variables are used for expressing cost function at hand for the individual problems as follows.

Largest Extension: A largest extension is one whose Hamming distance to the full set of arguments is the smallest. In the MaxSAT encoding, for each argument a we have the soft clause (x_a), stating that by leaving a out of the extension sought for, unit cost is incurred.

Repair: We express the Hamming distance to the extensions nearest to S by having for each argument $a \in S$ the soft clause (x_a) and for each argument $a \in A \setminus S$ the soft clause $(\neg x_a)$.

Adjust: The soft clauses are the same as for Repair, substituting S by E_0. However, we also have the additional hard clause (x_t) (if $t \in A \setminus E_0$) or $(\neg x_t)$ (if $t \in E_0$).

5. Empirical Evaluation

We implemented the SAT and MaxSAT approaches described in this work. In this section we present the results of an extensive evaluation of the different techniques for all three problems. All experiments were run on 2.83-GHz Intel Xeon E5440 quad-core machines with 32-GB RAM. For each instance, a timeout of 900 seconds was enforced. Our implementation and benchmark, as well as more detailed results, are available at `http://www.cs.helsinki.fi/group/coreo/comma18/`.

For the experiments, we used MiniSAT (version 2.2.0) [30] as a standard choice of a SAT solver in the iterative approaches. For generating the cardinality networks, we used the functionality offered by MaxPre [31]. For solving the MaxSAT encodings, we consider the following state-of-the-art SAT-based MaxSAT solvers using their default configurations: MaxHS (2.9.0) [32], Maxino (kdyn) [33], MSCG (2014) [34], OpenWBO (1.3.1) [35], and QMaxSAT (14.04) [36]. QMaxSAT implements a model-guided approach, using a SAT solver to iteratively find better solutions. Maxino, MSCG, and OpenWBO implement different variants of the popular core-guided approach, consisting of iteratively extracting unsatisfiable subsets (cores) of soft clauses using a SAT solver and compiling the cores into the working formula at each iteration via cardinality constraints modifications to the formula. MaxHS, on the other hand, implements the so-called implicit hitting set approach, iteratively using SAT solver for core extraction and an integer programming solver to find hitting sets over the accumulated set of cores, without changing the input instance, in contrast to the other approaches.

5.1. Benchmarks

We used the ICCMA 2017 benchmarks, described in more detail in [28].

Largest Extension. The instances are the sets A and B, each comprising of 350 AFs. The set A is used for admissible and complete semantics and set B for stable.

Repair. For each base AF, nine repair queries were generated by picking random sets of arguments of size 0.1, 0.3 and 0.5, three sets of each relative size. Thus there were 3150 instances for each admissible/complete and stable queries.

Adjust. A total of 1085 and 735 instances were generated from the set A for admissible and complete semantics, respectively, and 520 instances from the set B for stable semantics. As each adjust query contains an extension, the number of instances is constrained by our ability to find extensions. For each base AF for which five or more extensions were found by enumerating under a time limit of 15 minutes, five extensions were selected at random, and a query argument was selected outside the query extension from the union of all extensions. This ensured that all queries were satisfiable.

5.2. CNF Generation

When generating the propositional encodings, we use the following observations in our implementation.

- Self-attacks: If $(a,a) \in R$ for $a \in A$, then $x_a = 0$. Therefore we do not generate the clause $\neg x_a \vee \neg x_b$ encoding conflict-freeness for any $b \in A$, nor the clauses encoding admissibility where x_a is the antecedent, and instead generate the unit clause $(\neg x_a)$.
- Self-defence: If $(a,b),(b,a) \in R$, then for any extension E with $a \in E$, we know that a defends itself against $b \in A \setminus E$ (and vice versa). Therefore we do not generate the clauses encoding "admissibility" related to a and b.

Especially for very dense frameworks, such as those in the ABA2AF (filename *afinput*) family of instances in ICCMA 2017, arising from translating assumption-based argumentation frameworks into AFs [29], these simple observations can be very helpful in practice. We will provide empirical data supporting this idea in Section 5.4.

5.3. Results

We give an overview of the results by discussing a selection of combinations of the problem and the choice of semantics; more detailed results are available in the online supplement. An overview of the relative efficiency of all of the considered approaches is provided in Figure 2 (Largest Extension problem under admissible semantics), Figure 3 (Repair under stable semantics), and Figure 4 (Adjust under complete semantics), illustrating for each solver the number of instances solved (x-axis) under a specific per-instance time limit (y-axis). For the iterative SAT-based approaches, here stepping, bsearch, and prog refer to the Stepping, Binary, and Progressive iteration strategies, respectively, and CN and SC refer to cardinality network and sequential counter encodings, respectively.

Generally the MaxSAT approach seems the strongest, as a MaxSAT solver solved the most instances in each of the considered problem/semantics combinations except for Adjust under admissible semantics, where CN prog solved all instances while QMaxSAT timed out in one. The comparison between QMaxSAT and CN prog in this problem/semantics combination is shown in the right of Figure 5. The best MaxSAT solver for each combination varies. QMaxSAT was able to solve the most instances in two of the combinations: in Largest Extension under admissible semantics and in Repair under both admissible and complete semantics. QMaxSAT also tied for first place in Adjust under complete and stable semantics and was the fastest approach. QMaxSAT was also robust, being among the top approaches in every problem/semantic combination unlike any other considered approach. Each of Maxino, MSCG and MaxHS solved most instances in one

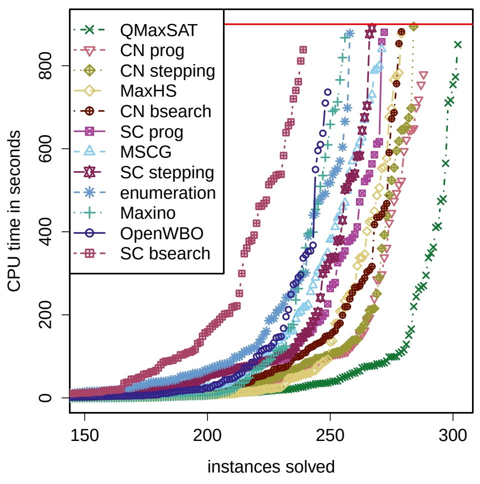

Figure 2. Largest Extension, admissible semantics

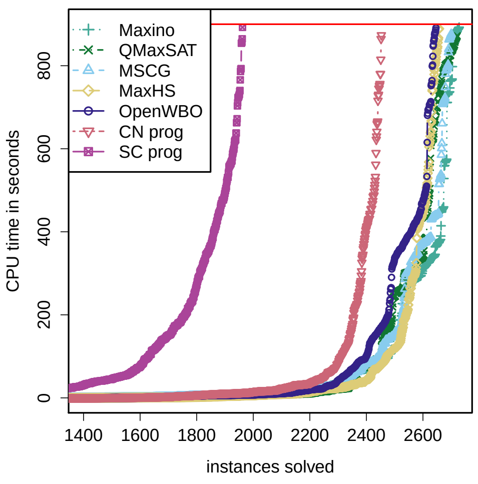

Figure 3. Repair, stable semantics

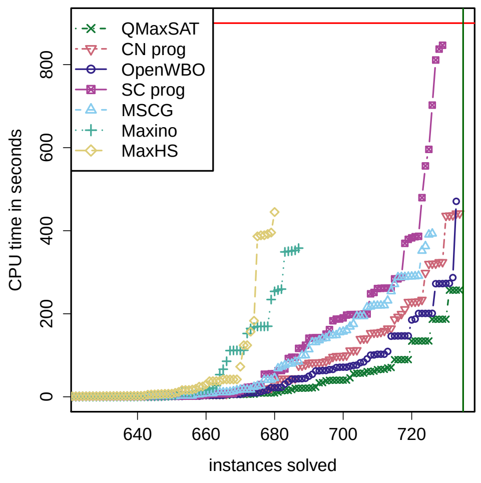

Figure 4. Adjust, complete semantics

problem/semantics pair. Maxino did so in Repair under complete semantics, MaxHS in Largest Extension under stable semantics, and MSCG narrowly beat QMaxSAT with one more solved instance in Repair under admissible semantics.

In some cases iterative strategies outperform MaxSAT. CN prog is competitive in Adjust in particular, being the only approach able to solve all instances under admissible for Adjust. Under complete semantics it tied with QMaxSAT for first place (CN prog had over two times larger cumulative running time, however). Moreover, in Largest Extension under admissible semantics, CN prog and CN stepping both solved more instances than any other MaxSAT solver except for the winner, QMaxSAT.

A closer look at the relative performance of the best iterative SAT-based and the best MaxSAT approach for a specific (problem, semantics) pair is provided in Figure 5 for Repair under stable semantics (left, Maxino (MaxSAT) vs CN prog) and for Adjust under admissible semantics (right, QMaxSAT vs CN prog). While the MaxSAT approach

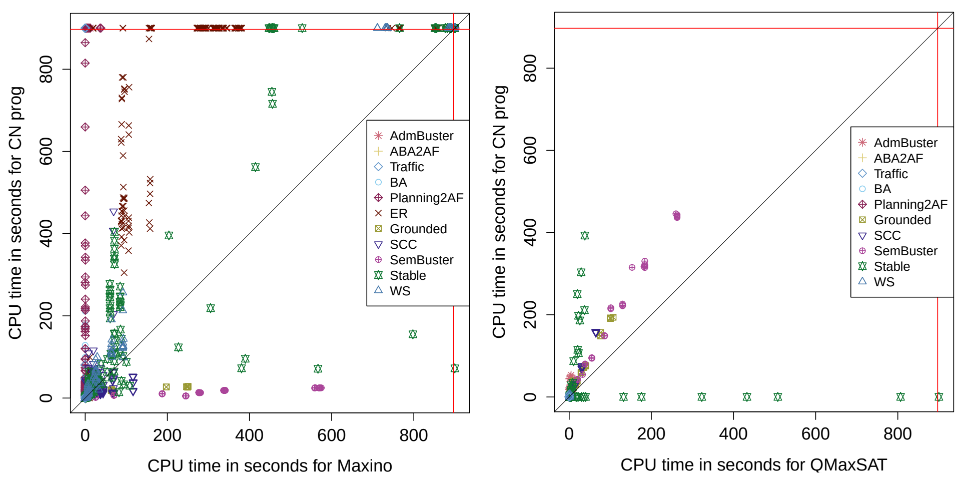

Figure 5. Best MaxSAT solver vs iterative SAT approach. Left: Repair stable, Maxino (MaxSAT) vs CN prog (progressive iterative SAT using cardinality networks). Right: Adjust admissible, QMaxSAT vs CN prog.

is clearly more effective overall, we observe that for specific families of instances, especially on the instance family *Sembuster* for Maxino and the instance family *Stable* for both Maxino and QMaxSAT, there are a significant number of instances on which the progressive iterative SAT approach performs better.

Overall, considering all of the problems and semantics, the model-guided approach as implemented in QMaxSAT would seem to the best single choice for solving these problems, exhibiting consistently good performance. Among the iterative SAT-based approaches, the choice of the applied cardinality encoding clearly matters, potentially due the fact that cardinality networks enable a fully incremental approach.

5.4. *Effects of CNF Generation*

Finally, we return to the effects of the revised CNF generation approach we applied in these experiments (recall Section 5.2). Compared to the standard CNF generation approach, the revised approach allows for not generating some of the clauses required for enforcing conflict-freeness and admissibility based on local structures in AFs that occur intuitively very frequently in dense AFs. One such family of very dense AFs is that of ABA2AF in the ICCMA 2017 benchmarks, totalling at 32 AFs.

Table 1 shows the cumulative effects of applying the revised CNF generation approach on the ABA2AF AFs. Compared to the standard approach, we observe a noticeable positive effect on both the size of the generated CNF formulas and the time spent on generating them: on average, the generated CNFs are 1.0% of the size of the CNFs generated with the the standard approach, and take only 35.7% of the time to be generated compared to the time needed when using the standard approach. Beyond the generation aspects, we also observed a noticeable effect on solver running times: on the Largest Extension problem, the cumulative running time of QMaxSAT after the revised approach was 1.2% of that after using the standard approach.

Table 1. Effects of revised CNF generation (recall Section 5.2) on the dense 32 AFs in the ABA2AF ICCMA 2017 benchmark family: cumulative size of the CNF instances (in megabytes), generation times in seconds (not including possible writing to disk), and solving times for Largest Extension using QMaxSAT in seconds.

Generation approach	**CNF size** (Mb)	**Generation time** (s)	**Solving time** (s)
Standard	66303.9	95.4	989.3
Revised	689.5	34.1	11.9

6. Conclusions

We presented a first computational study of applying SAT-based approaches to the dynamic problems of adjusting and repairing extensions in terms of a given AF, and further evaluated the approaches also on the problem of computing a largest extension. The results indicate that, while the performance on individual solvers can vary even noticeably depending on the underlying problem domain and semantics, MaxSAT currently offers effective tools to solving these problems. Going beyond the problems focused on in this paper, we showed that taking the local structure of an AF at hand into account within the process of generating the propositional encoding of the argumentation semantics can have a noticeable positive impact in terms of the generation time, encoding size, as well as the subsequent solving time, which we believe to be of interest more generally for improving the overall efficiency of SAT-based systems for argumentation.

Acknowledgements

This work was financially supported by Academy of Finland (grants 276412 and 312662) and DoCS Doctoral Programme in Computer Science of the University of Helsinki.

References

[1] T. Bench-Capon and P. E. Dunne, “Argumentation in artificial intelligence,” *Artificial Intelligence*, vol. 171, no. 10-15, pp. 619–641, 2007.

[2] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. R. Simari, M. Thimm, and S. Villata, “Towards artificial argumentation,” *AI Magazine*, vol. 38, no. 3, pp. 25–36, 2017.

[3] P. M. Dung, “On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games,” *Artif. Intell.*, vol. 77, no. 2, pp. 321–358, 1995.

[4] G. Charwat, W. Dvorák, S. A. Gaggl, J. P. Wallner, and S. Woltran, “Methods for solving reasoning problems in abstract argumentation - A survey,” *Artif. Intell.*, vol. 220, pp. 28–63, 2015.

[5] M. Thimm and S. Villata, “The first international competition on computational models of argumentation: Results and analysis,” *Artif. Intell.*, vol. 252, pp. 267–294, 2017.

[6] C. Cayrol, F. D. de Saint-Cyr, and M. Lagasquie-Schiex, “Change in abstract argumentation frameworks: Adding an argument,” *J. Artif. Intell. Res.*, vol. 38, pp. 49–84, 2010.

[7] R. Baumann, “Normal and strong expansion equivalence for argumentation frameworks,” *Artif. Intell.*, vol. 193, pp. 18–44, 2012.

[8] R. Booth, S. Kaci, T. Rienstra, and L. W. N. van der Torre, “A logical theory about dynamics in abstract argumentation,” in *Proc. SUM*, vol. 8078 of *LNCS*, pp. 148–161, Springer, 2013.

[9] S. Coste-Marquis, S. Konieczny, J. Mailly, and P. Marquis, “A translation-based approach for revision of argumentation frameworks,” in *Proc. JELIA*, vol. 8761 of *LNCS*, pp. 397–411, Springer, 2014.

[10] S. Coste-Marquis, S. Konieczny, J. Mailly, and P. Marquis, “On the revision of argumentation systems: Minimal change of arguments statuses,” in *Proc. KR*, pp. 52–61, AAAI Press, 2014.

[11] M. Diller, A. Haret, T. Linsbichler, S. Rümmele, and S. Woltran, “An extension-based approach to belief revision in abstract argumentation,” in *Proc. IJCAI*, pp. 2926–2932, AAAI Press, 2015.

[12] R. Baumann and G. Brewka, “AGM meets abstract argumentation: Expansion and revision for Dung frameworks,” in *Proc. IJCAI*, pp. 2734–2740, AAAI Press, 2015.

[13] R. Baumann, “What does it take to enforce an argument? Minimal change in abstract argumentation,” in *Proc. ECAI*, vol. 242 of *FAIA*, pp. 127–132, IOS Press, 2012.

[14] P. Bisquert, C. Cayrol, F. D. de Saint-Cyr, and M. Lagasquie-Schiex, “Enforcement in argumentation is a kind of update,” in *Proc. SUM*, vol. 8078 of *LNCS*, pp. 30–43, Springer, 2013.

[15] S. Doutre, A. Herzig, and L. Perrussel, “A dynamic logic framework for abstract argumentation,” in *Proc. KR*, pp. 62–71, AAAI Press, 2014.

[16] S. Coste-Marquis, S. Konieczny, J. Mailly, and P. Marquis, “Extension enforcement in abstract argumentation as an optimization problem,” in *Proc. IJCAI*, pp. 2876–2882, AAAI Press, 2015.

[17] J. P. Wallner, A. Niskanen, and M. Järvisalo, “Complexity results and algorithms for extension enforcement in abstract argumentation,” *J. Artif. Intell. Res.*, vol. 60, pp. 1–40, 2017.

[18] E. J. Kim, S. Ordyniak, and S. Szeider, “The complexity of repairing, adjusting, and aggregating of extensions in abstract argumentation,” in *TAFA 2013 Revised Selected Papers*, vol. 8306 of *LNCS*, pp. 158–175, Springer, 2014.

[19] R. Booth, M. Caminada, M. Podlaszewski, and I. Rahwan, “Quantifying disagreement in argument-based reasoning,” in *Proc. AAMAS*, pp. 493–500, IFAAMAS, 2012.

[20] W. Dvorák, M. Järvisalo, J. P. Wallner, and S. Woltran, “Complexity-sensitive decision procedures for abstract argumentation,” *Artif. Intell.*, vol. 206, pp. 53–78, 2014.

[21] F. Cerutti, M. Giacomin, and M. Vallati, “ArgSemSAT: Solving argumentation problems using SAT,” in *Proc. COMMA*, pp. 455–456, 2014.

[22] A. Niskanen, J. P. Wallner, and M. Järvisalo, “Optimal status enforcement in abstract argumentation,” in *Proc. IJCAI*, pp. 1216–1222, IJCAI/AAAI Press, 2016.

[23] W. Faber, M. Vallati, F. Cerutti, and M. Giacomin, “Solving set optimization problems by cardinality optimization with an application to argumentation,” in *Proc. ECAI*, vol. 285 of *FAIA*, pp. 966–973, IOS Press, 2016.

[24] P. Baroni, M. Caminada, and M. Giacomin, “An introduction to argumentation semantics,” *Knowledge Eng. Review*, vol. 26, no. 4, pp. 365–410, 2011.

[25] P. Besnard and S. Doutre, “Checking the acceptability of a set of arguments,” in *Proc. NMR*, pp. 59–64, 2004.

[26] C. Sinz, “Towards an optimal CNF encoding of Boolean cardinality constraints,” in *Proc. CP*, vol. 3709 of *LNCS*, pp. 827–831, Springer, 2005.

[27] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, “Cardinality networks: a theoretical and empirical study,” *Constraints*, vol. 16, no. 2, pp. 195–221, 2011.

[28] S. A. Gaggl, T. Linsbichler, M. Maratea, and S. Woltran, “Benchmark selection at ICCMA’17,” 2018. `http://argumentationcompetition.org/2017/benchmark_selection_iccma2017.pdf`.

[29] T. Lehtonen, J. P. Wallner, and M. Järvisalo, “From structured to abstract argumentation: Assumption-based acceptance via AF reasoning,” in *Proc. ECSQARU*, vol. 10369 of *LNCS*, pp. 57–68, Springer, 2017.

[30] N. Eén and N. Sörensson, “An extensible SAT-solver,” in *SAT 2003 Selected Revised Papers*, vol. 2919 of *LNCS*, pp. 502–518, Springer, 2004.

[31] T. Korhonen, J. Berg, P. Saikko, and M. Järvisalo, “MaxPre: An extended MaxSAT preprocessor,” in *Proc. SAT*, vol. 10491 of *LNCS*, pp. 449–456, Springer, 2017.

[32] J. Davies and F. Bacchus, “Exploiting the power of MIP solvers in MAXSAT,” in *Proc. SAT*, vol. 7962 of *LNCS*, pp. 166–181, Springer, 2013.

[33] M. Alviano, C. Dodaro, and F. Ricca, “A MaxSAT algorithm using cardinality constraints of bounded size,” in *Proc. IJCAI*, pp. 2677–2683, AAAI Press, 2015.

[34] A. Morgado, A. Ignatiev, and J. Marques-Silva, “MSCG: Robust core-guided MaxSAT solving,” *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 9, pp. 129–134, 2014.

[35] R. Martins, V. M. Manquinho, and I. Lynce, “Open-WBO: A modular MaxSAT solver,,” in *Proc. SAT*, vol. 8561 of *LNCS*, pp. 438–445, Springer, 2014.

[36] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa, “QMaxSAT: A partial Max-SAT solver,” *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 8, no. 1/2, pp. 95–100, 2012.