ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1652

A New Stochastic Local Search Approach for Computing Preferred Extensions of Abstract Argumentation

Dangdang Niu[1] and **Lei Liu**[1] and **Shuai Lü**[1,2,3+]

Abstract. In this paper, we proposed a new stochastic local search algorithm Inc-CCA$_{EP}$ for computing the preferred extensions in (abstract) argumentation frameworks (AF). Inc-CCA$_{EP}$ realizes an incremental version of Swcca, specially designed for computing the preferred extensions in AF. Experiments show that, Inc-CCA$_{EP}$ notably outperforms the state-of-the-art solvers consistently on random benchmarks with non-empty preferred extensions.

1 Introduction

Dung's theory of (abstract) argumentation frameworks (AF) provides a general model for computational argumentation [1]. For preferred semantics in AF, CEGARTIX [2] and ArgSemSAT [3] are two representative SAT-based argumentation systems, which both rely on iteratively calling to complete SAT solvers. Each of above two systems is ranked 1st or 2nd in the SE-PR, DS-PR and EE-PR tracks of the first International Competition on Computational Models of Argumentation (ICCMA'15).

There are two popular kinds of algorithms for SAT: conflict driven clause learning, and stochastic local search (SLS). The efficiency of SLS algorithms mostly depend on the heuristic methods selected by them. An efficient heuristic method named configuration checking (CC) [4] has been proposed. Then Swcca (smoothed weighting and configuration with aspiration) [5] is designed based on the heuristics named configuration checking with aspiration (CCA) which is an improvement of CC.

Only complete SAT solvers have been exploited for computing preferred extensions in AF so far. It is a natural question that how the appealing SLS approaches could advance the performance. The aim of this paper is to answer above question. And a novel approach Inc-CCA$_{EP}$ is proposed based on the SLS algorithm of SAT. This paper is organized as follows. Section 2 recalls the basic concepts of AF and CCA. Section 3 introduces our SLS algorithms for computing preferred extensions, while Section 4 describes the test setting and comments the experimental results. Section 5 concludes the paper.

2 Preliminaries

An AF is a pair $F = (A, R)$ where A is a set of arguments and $R \subseteq A \times A$ is the attack relation. An extension $S \subseteq A$ is *conflict–free* iff $\nexists a, b \in S$ s.t. $a \rightarrow b$. An argument $a \in A$ is *acceptable* with respect to a set $S \subseteq A$ iff $\forall b \in A$ s.t. $b \rightarrow a$, $\exists c \in S$ s.t. $c \rightarrow b$. $S \subseteq A$ is *admissible* iff S is *conflict–free* and every element of S is *acceptable* with respect to S. $S \subseteq A$ is an *admissible extension* iff S is *admissible*. $S \subseteq A$ is a *preferred extension* iff S is a maximal *admissible extension*.

Given an AF $F = (A, R)$, a key problem of instantiating the SAT-based framework is how the AF reasoning tasks are encoded as CNF formulae. Let $\varphi_{adm}(F)$ be the CNF formula which corresponds to the expression of admissible semantics of F, the set of all variables appear in $\varphi_{adm}(F)$ is in correspondence with A. $\varphi_{adm}(F)$ is shown as follows [3]:

$$\varphi_{adm}(F) = \bigwedge_{(b,a)\in R}((\neg v_a \vee \neg v_b) \wedge (\neg v_a \vee \bigvee_{(c,b)\in R} v_c)) \quad (1)$$

Given a model P, let $\alpha(P)$ be the set of positive literals in P. Then, a model P of $\varphi_{adm}(F)$ is in correspondence with a preferred extension of F iff there is no model Q with $\alpha(P) \subset \alpha(Q)$.

Let $V(F)$ be the set of all variables appear in the CNF formula F. $N(x) = \{y \mid y \in V(F)$ and y occurs in at least one clause with $x\}$ is the set of all neighboring variables of a variable x. Then the configuration of a variable $v \in V(F)$ is a vector C_v consisting of truth values of all variables in $N(v)$ under the current assignment s.

In the implementation of CCA, each clause $c \in \Phi$ is associated with a positive integer number $w(c)$ as its weight, in which w is a weighted formula. $cost(\Phi, s)$ denotes the total weight of all unsatisfied clauses under the assignment s. Let $score(v) = cost(F, s) - cost(F, s')$, measuring the benefit of flipping v, where s' is obtained from s by flipping v. Any element in the array *confChange* is an indicator for a variable. *confChange*$[v] = 1$ means the configuration of variable v has been changed since v's last flip; and *confChange*$[v] = 0$ on the contrary.

CCA heuristics: A configuration changed decreasing (CCD) variable v is a variable with both *confChange*$[v] = 1$ and $score(v) > 0$. A significant decreasing (SD) variable v is a variable with $score(v) > g$, where g is a positive integer large enough. In the literature [5], g is set to the averaged clause weight (over all clauses). CCA selects CCD variable with greatest score to flip firstly. The SD variable with the greatest score is selected to flip if there are no CCD variables. If there are neither CCD variables nor SD variables, CCA switches to diversification mode [5].

3 Inc-CCA$_{EP}$

We are now in a position to introduce our proposed procedure Inc-CCA$_{EP}$ which is listed in Algorithm 1.

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China, email: ddniu15@mails.jlu.edu.cn, {liulei, lus}@jlu.edu.cn.
[2] College of Mathematics, Jilin University, Changchun 130012, China.
[3] Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China.

Alogrithm 1 Inc-CCA$_{EP}$

1: **Input**: $F = (A, R)$
2: **output**: $E_p \subseteq 2^A$
3: $E_p = \emptyset$
4: $\Phi_0 = \varphi_{adm}(F)$
5: (Φ, M) = unit-propagation(Φ_0)
6: $s \leftarrow$ randomly generated truth assignment
7: initialize heuristic information
8: **for** *step* $\leftarrow$ 1 **to** *maxsteps* **do**
9: **if** s satisfies Φ
10: **while** $\exists \neg v_i \in s$ & $s|_{\text{flip}(vi)}$ satisfies Φ
11: $s = s|_{\text{flip}(vi)}$
12: update heuristic information
13: $E_p = E_p \cup \{s\}$
14: $\Phi = \Phi \wedge \bigvee_{\neg vi \in s} v_i$
15: add heuristic information about $\bigvee_{\neg vi \in s} v_i$
16: v = CCA()
17: $s = s|_{\text{flip}(v)}$
18: update heuristic information
19: E_p = merge(E_p, M)
20: **return** E_p

The main idea behind Inc-CCA$_{EP}$ is that it intends to compute as many as possible admissible extensions based on $\varphi_{adm}(F)$ and then finding all preferred extensions. In order to efficiently compute preferred extensions, we design two strategies in Algorithm 1:

Strategy 1. There may be some unit clauses in $\varphi_{adm}(F)$. So we use unit propagation to reduce the scale of $\varphi_{adm}(F)$ in Line 5.

Strategy 2. After Swcca searches a model T, if T is still a model after flipping some negative literals in it, we flip them in Line 10-11. This strategy can reduce the calling times of Swcca.

In Algorithm 1, the heuristic information appeared in Inc-CCA$_{EP}$ mainly concludes the scores of all variables, the weights of all clauses and the flipping time stamps of all variables [5]. Since Inc-CCA$_{EP}$ is an incomplete algorithm, it does not know whether all possible preferred extensions have been found or not. So, we use *maxsteps* to control the end of Inc-CCA$_{EP}$ in Line 8. In order to avoid searching an admissible extension which is the subset of any admissible extension searched before, we add $\bigvee_{\neg vi \in s} v_i$ to Φ in Line 14. It employs CCA heuristics to pick flipping variable in Line 16. In Line 19, merge function is used to add all literals in M to each model in E_p and then find all possible preferred extensions.

4 Experimental results

Our experiments are conducted on the PC with a quad-core Intel(R) Core(TM) i7-3700, 8GByte RAM and Ubuntu 14.04 operating system. Because Inc-CCA$_{EP}$ cannot finish their search processes automatically even if they have found all preferred extensions, we use actually searching time to measure their efficiencies. We do the experiments on randomly generated AFs by *probo*[4] which is used in ICCMA'15. We input two parameters $|A|$ and p to *probo*. $|A|$ is the number of arguments, and p is the probability that there is an attack for each ordered pair of arguments (self-attacks are include). CEGARTIX[5] and ArgSemSAT[6] are two alternative AF reasoners in our experiments.

[4] https://sourceforge.net/p/probo/code/HEAD/tree/trunk/doc/
[5] http://www.dbai.tuwien.ac.at/research/project/argumentation/cegartix/
[6] http://sourceforge.net/projects/argsemsat/

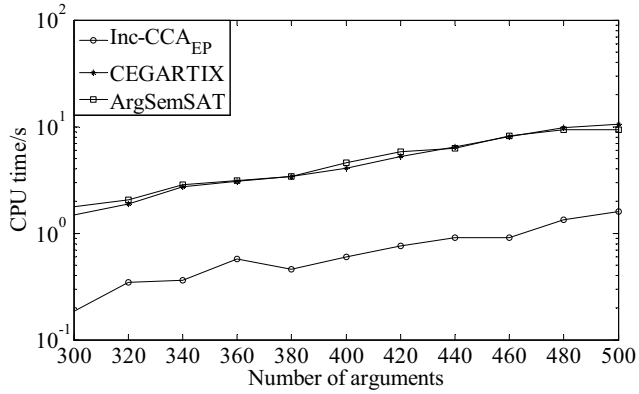

Figure 1. Experiments on random AF instances with fixing $p = 0.5$.

We generate AF instances with fixing $p = 0.5$, and $|A|$ is ranging from 360 to 500 with a step of 20. All instances have non-empty preferred extensions. The test results are given in Figure 1.

In Figure 1, we can see that the actual searching time of Inc-CCA$_{EP}$ is far less than the executing time of CEGARTIX and ArgSemSAT. When fixing $p = 0.5$, the efficiency of Inc-CCA$_{EP}$ is only about a tenth of the efficiency of CEGARTIX or ArgSemSAT. ArgSemSAT is a little better than CEGARTIX. The reason is that CEGARTIX is more suitable for hard AF instances.

5 Conclusions

We innovatively exploit stochastic local search for enumerating preferred extensions in abstract argumentation in this paper. And Inc-CCA$_{EP}$ is proposed based on Swcca for enumerating preferred extensions in abstract argumentation. Experimental results show that Inc-CCA$_{EP}$ significantly outperform existing systems on random AF instances with non-empty preferred extensions.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China under Grant No. 61300049; the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20120061120059; and the Natural Science Research Foundation of Jilin Province of China under Grant Nos. 20140520069JH, 20150101054JC.

REFERENCES

[1] P. M. Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence*, 77(2), 321-358, (1995).
[2] W. Dvořák, M. Järvisalo, J. P. Wallner and S. Woltran, 'Complexity-sensitive decision procedures for abstract argumentation', *Artificial Intelligence*, 206, 53-78, (2014).
[3] F. Cerutti, M. Giacomin and M. Vallati, 'ArgSemSAT: Solving argumentation problems using SAT', in *Proc. 5th International Conference on Computational Models of Argument (COMMA)*, pp. 455-456, Scottish Highlands, UK, (2014).
[4] S. Cai, K. Su and A. Sattar, 'Local search with edge weighting and configuration checking heuristics for minimum vertex cover', *Artificial Intelligence*, 175(9-10), 1672-1696, (2011).
[5] S. Cai and K. Su, 'Configuration checking with aspiration in local search for SAT', in *Proc. 26th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 434-440, Toronto, Ontario, Canada, (2012).