Algorithmic Approaches to Computational Models of Argumentation

Matthias Thimm

Institute for Web Science and Technologies (WeST),
University of Koblenz-Landau, Germany

1 Introduction

Computational models of argumentation [1] are approaches for non-monotonic reasoning that focus on the interplay between arguments and counterarguments in order to reach conclusions. These approaches can be divided into either *abstract* or *structured* approaches. The former encompass the classical abstract argumentation frameworks following Dung [8] that model argumentation scenarios by directed graphs, where vertices represent arguments and directed links represent attacks between arguments. In these graphs one is usually interested in identifying *extensions*, i. e., sets of arguments that are mutually acceptable and thus provide a coherent perspective on the outcome of the argumentation. On the other hand, structured argumentation approaches consider arguments to be collections of formulas and/or rules which entail some conclusion. The most prominent structured approaches are ASPIC+ [16], ABA [22], DeLP [13], and *deductive argumentation* [2]. These approaches consider a knowledge base of formulas and/or rules as a starting point.

2 Algorithms for Abstract Argumentation

According to [6], algorithms for solving reasoning problems in abstract argumentation can generally be categorised into two classes: *reduction-based* approaches and *direct* approaches.

Reduction-based approaches such as ASPARTIX-D [10,12] and ArgSemSAT [5] translate the given problem for abstract argumentation—such as determining a single stable extension—into another formalism and use dedicated (and mature) systems for that formalism to solve the original problem. For example, ASPARTIX encodes the problem of finding a stable extension in abstract argumentation into the question of finding an answer set of an answer set program [14]. Due to the direct relationship of answer sets and stable models the answer set program only needs to model the semantics of the abstract argumentation framework in a faithful manner and represent the actual framework. ASPARTIX-D then makes use of the Potassco ASP solvers[1] to solve the reduced problem and translate their output back to the original question. Similarly, ArgSemSAT decodes the problem as a SAT instance and uses the Glucose[2] SAT

[1] `http://potassco.sourceforge.net`

[2] `http://www.labri.fr/perso/lsimon/glucose/`

solver to solve the latter. Internally, solvers such as the Potassco ASP solvers and SAT solvers make use of sophisticated search strategies such as *conflict-driven nogood learning* or *conflict-driven clause learning*, see [14,3] for details.

Direct approaches to solve reasoning problems in abstract argumentation are inspired by similar search strategies but directly realise these algorithms for abstract argumentation. For example, solvers such as ArgTools [17] and heureka [15] are based on the DPLL (Davis-Putnam-Logemann-Loveland) backtracking algorithm from SAT solving [3, Chapter 3]. Basically, they exhaustively explore the search space of all possible sets of arguments to determine, e. g., a stable extension but include various optimisations and specific search strategies to prune the search space as much as possible to keep runtime low. Another direct solver, EqArgSolver [18], uses a different approach though, and is inspired by an iteration scheme originally designed to solve problems for probabilistic argumentation [11]. For a more detailed discussion of the different approaches to solving problems in abstract argumentation see [4].

Recently, approximate methods for reasoning problems in abstract argumentation have been introduced as well [20]. The algorithms of [20] follow the paradigm of *stochastic local search*, i. e., incomplete optimisation algorithms that aim at reaching an optimal value of a target function by small random changes of the parameters, see e. g. [3, Chapter 6] for a deeper discussion in the context of solving the satisfiability problem (SAT). The core idea of these algorithms is as follows. Considering the labelling approach to the semantics of abstract argumentation frameworks, they start from a labelling that randomly assigns the acceptability status in and out to all arguments of the input argumentation framework. As long as this labelling is not stable—i. e. as long as the arguments labelled in do not form a stable extension—one mislabelled argument is selected and its acceptability status is flipped. Albeit being a simple idea it can outperform traditional algorithms, in particular on *random* instances with little structure.

3 Algorithms for Structured Argumentation

Queries are answered in structured argumentation approaches, e. g. in the case of ASPIC+ [16], by determining all arguments constructible from the knowledge base, identifying attacks between these arguments using e. g. contradictions between conclusions of different arguments, and resolving the conflicts by representing the constructed arguments and attacks as an abstract argumentation framework and relying on reasoning methods for this abstract case. Computationally, reasoning with structured argumentation approaches can be quite demanding as both checking whether a set of formulas and/or rules is an argument can be challenging and the number of arguments in a knowledge base may be super-polynomial (and even infinite in some approaches). Some formal analyses on this, in particular regarding the approach of ABA, can be found in [7,9]. Existing solvers for ASPIC+ that implement complete reasoning procedures are, e. g., TOAST [19] and EPR[3]. See [4] for a survey on sound and complete algorithms and implementations for structured argumentation approaches.

A recent approach [21] to approximate reasoning with structured argumentation is based on sampling of arguments, instead of constructing all possible arguments. There, two parametrised algorithms are developed that solve the general problem of checking whether a certain proposition is acceptable wrt. a given knowledge base. Both algorithms rely on *sampling* arguments in order to avoid enumerating all arguments of a knowledge base. The first algorithm $\textsc{rand}_I$ samples arguments *independently* by 1) selecting some rule from the knowledge base to be the top rule of the argument, and 2) recursively selecting rules where their conclusion appears in the body of a previously selected rule, until a valid argument is found. This process is repeated for a fixed number of arguments, yielding a set of arguments that is a subset of all possible arguments. The second algorithm $\textsc{rand}_D$ samples arguments *directionally* by 1) sampling some argument that has the query as conclusion, and 2) recursively sampling counterarguments of previously sampled arguments.

References

1. Atkinson, K., Baroni, P., Giacomin, M., Hunter, A., Prakken, H., Reed, C., Simari, G.R., Thimm, M., Villata, S.: Toward artificial argumentation. AI Magazine **38**(3), 25–36 (October 2017)
2. Besnard, P., Hunter, A.: Constructing argument graphs with deductive arguments: a tutorial. Argument & Computation **5**(1), 5–30 (2014)
3. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
4. Cerutti, F., Gaggl, S.A., Thimm, M., Wallner, J.P.: Foundations of implementations for formal argumentation. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 15. College Publications (February 2018), also appears in IfCoLog Journal of Logics and their Applications 4(8):2623–2706, October 2017
5. Cerutti, F., Giacomin, M., Vallati, M.: Argsemsat: Solving argumentation problems using SAT. In: Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014. pp. 455–456 (2014)
6. Charwat, G., Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - A survey. Artificial Intelligence **220**, 28–63 (2015)
7. Dimopoulos, Y., Nebel, B., Toni, F.: On the computational complexity of assumption-based argumentation for default reasoning. Artificial Intelligence **141**(1), 57 – 78 (2002)
8. 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), 321–358 (1995)
9. Dvořák, W., Dunne, P.E.: Computational problems in formal argumentation and their complexity. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.)

[3] `http://www.wietskevisser.nl/research/epr/`

Handbook of Formal Argumentation, chap. 14. College Publications (February 2018)

10. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Technical Report DBAI-TR-2008-62, Technische Universität Wien (2008)
11. Gabbay, D., Rodrigues, O.: A self-correcting iteration schema for argumentation networks. In: Proceedings of the Fifth International Conference on Computational Models of Argumentation (COMMA'14) (2014)
12. Gaggl, S.A., Manthey, N.: Aspartix-d: Asp argumentation reasoning tool - dresden. In: System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15). ArXiv (2015)
13. García, A.J., Simari, G.R.: Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers. Argument & Computation **5**(1), 63–88 (2014)
14. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012)
15. Geilen, N., Thimm, M.: Heureka - a general heuristic backtracking solver for abstract argumentation. In: Proceedings of the 2017 International Workshop on Theory and Applications of Formal Argument (TAFA'17) (August 2017)
16. Modgil, S., Prakken, H.: The ASPIC+ framework for structured argumentation: A tutorial. Argument & Computation **5**, 31–62 (2014)
17. Nofal, S., Atkinson, K., Dunne, P.E.: Looking-ahead in backtracking algorithms for abstract argumentation. International Journal of Approximate Reasoning **78**, 265–282 (2016)
18. Rodrigues, O.: A forward propagation algorithm for the computation of the semantics of argumentation frameworks. In: Theory and Applications of Formal Argumentation - 4th International Workshop, TAFA 2017, Melbourne, VIC, Australia, August 19-20, 2017, Revised Selected Papers. pp. 120–136 (2017)
19. Snaith, M., Reed, C.: TOAST: online aspic+ implementation. In: Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA 2012). pp. 509–510. IOS Press (2012)
20. Thimm, M.: Stochastic local search algorithms for abstract argumentation under stable semantics. In: Modgil, S., Budzynska, K., Lawrence, J. (eds.) Proceedings of the Seventh International Conference on Computational Models of Argumentation (COMMA'18). Frontiers in Artificial Intelligence and Applications, vol. 305, pp. 169–180. Warsaw, Poland (September 2018)
21. Thimm, M., Rienstra, T.: Approximate reasoning with aspic+ by argument sampling (2019), under review
22. Toni, F.: A tutorial on assumption-based argumentation. Argument & Computation **5**(1), 89–117 (2014)