

Enhanced SAT-based Argumentation Solvers Using *in* Label Priority Strategy and Large Language Model Tuning

Qi Liu

Hubei University of Technology
Wuhan, China
102401137@hbut.edu.cn

Guangcheng Dong

Hubei University of Technology
Wuhan, China
a257158281@126.com

Aoxu Ji

Hubei University of Technology
Wuhan, China
jiaoxu@hbut.edu.cn

Hang Ding

Hubei University of Technology
Wuhan, China
dinghang@hbut.edu.cn

Mao Luo

Hubei University of Technology
Wuhan, China
luomao@hbut.edu.cn

Ningning He

Hubei University of Technology
Wuhan, China
hnn102211103@163.com

Caiquan Xiong

Hubei University of Technology
Wuhan, China
xiongqcq@hbut.edu.cn

Xinyun Wu

Hubei University of Technology
Wuhan, China
xinyun@hbut.edu.cn

Yuanzhi Ke

Hubei University of Technology
Wuhan, China
keyuanzhi@hbut.edu.cn

Abstract—We present four solvers—ASSAT_inPriority, ASSAT_Amsat, ASSAT_Cadical and ASSAT_LLM to ICCMA 2025, all of which are improved versions of *ArgSemSAT*. Some solvers replace the original *Minisat* solver used in *ArgSemSAT*, and all solvers incorporate optimizations to the rephasing strategies of the SAT solver. All four solvers participate in the following subtracks of the Main and No-Limits track: SE-{PR, ST, SST}.

Index Terms—Abstract Argumentation, Solver

I. INTRODUCTION

ArgSemSAT is a well-established solver for abstract argumentation frameworks [1]. In particular, *ArgSemSAT* participated in the ICCMA 2017 and was the winner of the preferred semantics track, demonstrating its effectiveness and efficiency in computing argumentation semantics.

ArgSemSAT employs a encoding approach grounded in the three-valued labelling-based semantics [2]. In this method, each argument in the argumentation framework is assigned exactly one of three mutually exclusive labels:

- *in* (I_i): the argument is justified.
- *out* (O_i): the argument is overruled.
- *undec* (U_i): the argument is neither justified nor overruled due to uncertainty.

In our work, we observe that arguments assigned the *in* label typically lead to stronger propagation effects during the SAT solving process, compared to those assigned *out* or *undec* labels. Based on this observation, we adopt the rephasing strategy that prioritizes assigning *in*-label decision variables to true during the SAT solving process.

II. ARCHITECTURE

A. ASSAT_inPriority

In our previous work [3], our team proposed an *in*-label prioritizing variable branching strategy that significantly improved the SAT solver from two aspects: the initialization of the polarity values of decision variables and the evaluation criteria for variable's activity, enabling efficient solving of a preferred extension of argumentation frameworks. We further found that the *in*-variable-priority strategy can also be effectively applied in the search process of SAT solvers. In particular, it can be integrated into the rephasing strategy used after each restart, further enhancing the solver's performance and robustness.

Building upon these promising results, We retained the original *Minisat* solver used in *ArgSemSAT*, and based on our observation of the advantages associated with variables labeled as *in*, we modified the original variable polarity initialization strategy, as well as the rephasing strategy. Specifically, during the initial polarity assignment, variables corresponding to *in*-labeled arguments are initialized with negative polarity, whereas variables associated with other labels are initialized with positive polarity. Separately, in the phase saving process of *Minisat*, if a variable corresponds to an *in*-labeled argument, we assign its polarity with a 35% probability of being negative, a 15% probability of being positive, and a 50% probability of preserving its previous polarity. For variables associated with other labels, we assign 15% probability to both negative and positive polarities, while maintaining the previous polarity with a 70% probability. This probabilistic polarity strategy introduces a degree of diversification, and our

experimental results demonstrate that it significantly improves solving performance.

B. ASSAT_AmSAT

In this solver, we replace the original *Minisat* SAT solver of *ArgSemSAT* with the *AmSAT* SAT solver [4], which improves the rephasing strategy and significantly enhances solving efficiency. The *Relaxed CDCL Approach* [5] employed by *AmSAT* aims at preserving “promising” partial assignments. A partial assignment is considered “promising” if it satisfies two conditions: it does not lead to a conflict and it has a relatively long assignment trail.

Building on this, we further modified *AmSAT*’s rephasing strategy. Specifically, *AmSAT*’s original rephasing strategy randomly selects between two branches, `info_based` and `rand_based`, after each restart. We extended this approach to randomly select among three branches: `info_based`, `rand_based`, and `in_priority`. The implementation of `in_priority` is exactly the same as the rephasing strategy used in *ASSAT_inPriority*.

C. ASSAT_Cadical

In this solver, we replace the original the *Minisat* SAT solver of *ArgSemSAT* with the *Cadical* SAT solver. Motivated by our empirical observations regarding the propagation strength of `in`-labeled arguments, we configure the initial phase assignments such that variables corresponding to `in` labels are initialized to `true`, whereas all other variables are initialized to `false`. This heuristic notably improves the solver’s performance by guiding the search toward promising regions of the solution space.

D. ASSAT_LLM

In this solver, we integrate *ArgSemSAT* with large language models to enhance the branching decision process by associating the polarity selection of branching variables with the historical frequency of conflict triggers. Specifically, the solver prefers to explore the polarity direction that has led to fewer conflicts. Leveraging large language models, we generate diverse heuristic strategies and establish an automated framework for code generation and strategy evaluation. Through an evolutionary iteration strategy, we enable a directed exploration of the strategy space, ultimately identifying the optimal conflict-driven branching heuristic.

III. SUMMARY

We presented four SAT-based solvers designed for various reasoning tasks in abstract argumentation frameworks. These solvers integrate techniques including customized rephasing strategies based on `in`-labelled argument behaviors, relaxed backtracking for preserving promising partial assignments, and the use of large language models to automatically generate and refine branching heuristics via evolutionary search. Each solver targets a different aspect of search efficiency and solution quality. The source code for all solvers is available at The source code for all solvers is available at <https://github.com/36298liu/Solvers-for-ICCMA-2025>.

REFERENCES

- [1] F. Cerutti, M. Vallati, and M. Giacomini, “ArgSemSAT: Solving Argumentation Problems Using SAT,” in *Proc. 5th Int. Conf. on Computational Models of Argument (COMMA 2014)*, Frontiers in Artificial Intelligence and Applications, vol. 266, IOS Press, 2014, pp. 455–456.
- [2] S. Modgil and M. Caminada, “Proof theories and algorithms for abstract argumentation frameworks,” in *Argumentation in Artificial Intelligence*, I. Rahwan and G. R. Simari, Eds. Boston, MA: Springer, 2009, pp. 105–129.
- [3] M. Luo, J. Xiong, N. He, C. Xiong, X. Wu, and J. Wu, “An in-label prioritizing variable branching strategy of SAT solvers for a preferred extension of argumentation frameworks,” in *PRICAI 2024: Trends in Artificial Intelligence*, vol. 5, Springer, 2025, pp. 216–231.
- [4] S. Li, C.-M. Li, M. Luo, J. Coll, S. Cherif, D. Habet, and F. Manyà, “ESA Solvers, Kissat MAB Binary and AMSAT in SAT competition 2023,” in *Proc. of SAT Competition 2023 – Solver and Benchmark Descriptions*, 2023, pp. 32–33.
- [5] S. Cai and X. Zhang, “Four Relaxed CDCL Solvers,” in *Proceedings of SAT Race 2019: Solver and Benchmark Descriptions*, University of Helsinki, 2019, pp. 35–38.