Project Proposal: Efficient Neural Clause Selection by Weight*

Filip Bártek and Martin Suda

Czech Technical University in Prague
filip.bartek@cvut.cz

1 Introduction

Saturation-based automated theorem provers (ATPs) that use the given clause algorithm [1] maintain two sets of clauses during the proof search: processed and unprocessed. In each iteration of the saturation loop, the prover selects a clause, called the given clause by convention, from the unprocessed set. All possible inferences are then made between the given clause and the processed clauses. The newly inferred clauses are added to the unprocessed set, while the given clause is moved from the unprocessed set to the processed set. When the proof search successfully concludes (by inferring the empty clause, a trivial contradiction), a subset of processed clauses constitutes the proof.

The basic clause selection heuristics choose the clause that is the oldest according to the time it was inferred, or the smallest in the symbol count, that is, the number of symbol occurrences [11]. The symbol count heuristic can be generalized by assigning each symbol a weight (a positive real number); the clause weight is then calculated by summing the weights of the symbol occurrences. In addition to that, clauses can be penalized, for example, for each positive literal, maximal literal, or unorientable equation [10].

The goal of this project is to improve the standard weight-based clause selection heuristic by machine learning from proof searches. Specifically, our aim is to find problem-specific values for the coefficients of the clause weight function (that is, the weights of symbols, variables, maximal literals, unorientable equations, etc.). These parameter values are to be output from a neural network (NN) trained on successful proof searches.

In section 2 we describe in detail our proposed solution in its basic form – a system that automatically configures the parameters of the clause weight function in a problem-specific fashion. In section 3 we outline possible generalizations and modifications.

2 GNN-based clause selection

Prediction When a new problem is to be solved, the problem is processed with a graph neural network (GNN) [13, 8, 4, 9]. The GNN produces a weight w_f for each symbol f and a common weight w_X for all variables. The weights are then used to instantiate a weight-based clause selection heuristic: The weight of a clause is computed as the sum of the weights of all the symbol and variable occurrences in the clause.

Note that our GNN only needs to be evaluated once at the beginning of the proof search: Once the symbol and variable weights are calculated, they can be used throughout the proof search without any additional input from the GNN. This can be interpreted as a conservative

*This work is supported by the Czech Science Foundation project no. 20-06390Y (JUNIOR grant), and the Grant Agency of the Czech Technical University in Prague, grant no. SGS20/215/OHK3/3T/37.

modification of the standard clause selection heuristic: We introduce neural guidance into the proof search without increasing the cost of computing the weight of a newly inferred clause. This contrasts with neural proof guidance systems that process each clause considered for selection with a NN, such as ENIGMA Anonymous [5] or the prototype extension of E by Loos et al. [7].

Training The GNN is trained on successful proof searches. Similarly to ENIGMA [5], our GNN is trained so that the clause selection heuristic favors clauses that have been observed to contribute to a proof. Using a GNN that does not have access to symbol names, we train a signature-agnostic system.

Each training example consists of an input problem, a proof clause c^+, and a non-proof clause c^-. The loss function ℓ is defined so that it is monotone with respect to the weight of the proof clause $w(c^+)$ and anti-monotone with respect to the weight of the non-proof clause $w(c^-)$. Additional penalty terms, scaled by hyperparameters λ_f and λ_X, ensure that the symbol weights w_{f_i} and the variable weight w_X do not drift far from the standard value 1. The loss of the proof clause c^+ and the non-proof clause c^- in a problem with symbols $f_1, \ldots, f_n$ is:

$$\ell(c^+, c^-) = -\log \mathrm{sigmoid}(w(c^-) - w(c^+)) + \lambda_f \frac{1}{n} \sum_{i=1}^{n} (\log w_{f_i})^2 + \lambda_X (\log w_X)^2 \tag{1}$$

This loss design is inspired by the approach that we successfully applied to symbol precedence recommendation [2].

Training examples will be collected by running the target ATP Vampire [6] on problems randomly sampled from the TPTP problem library [12].

3 Possible modifications

Additional input features The clause weight function may include additional clause features available in the prover, namely clause derivation depth and size. These clause features may be multiplied by coefficients predicted by the GNN to make the influence of the features trainable. In a similar fashion, the weights of terms, atoms, and literals may be augmented by term-ordering-aware features [11], namely literal maximality in the clause, term maximality in an equation, and orientability of an equation.

Binary cross-entropy loss A straightforward alternative to the loss function defined above (1) is provided by training a binary clause classifier that predicts whether an input clause is non-proof. If we allow the symbol and variable weights to span all real numbers, we can train the NN using the standard binary cross-entropy loss, which allows extracting a prediction of probability of the input clause classifying as non-proof. Ranking the clauses by this probability prioritizes the inference of clauses that are likely to contribute to the proof. Although allowing negative term weights forfeits fairness of the clause selection, this can be salvaged (as is anyway done in the typical clause selection setup) by alternating weight-based and age-based clause section under some ratio.

Recursive neural network Without increasing the asymptotic computational cost of the evaluation of the clause weight, we can train a recursive neural network (RNN) for the clause syntax directed acyclic graph (dag) [3]. Depending on the complexity of the RNN, this can significantly increase the computational cost of evaluating the weight of a clause. The RNN may use a GNN to calculate the initial embeddings of the symbols and variables.

References

[1] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 19–99. Elsevier and MIT Press, 2001. ISBN 0-444-50813-9. doi:10.1016/b978-044450813-3/50004-7.

[2] Filip Bártek and Martin Suda. Neural precedence recommender. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2021. ISBN 978-3-030-79875-8. doi:10.1007/978-3-030-79876-5_30.

[3] Karel Chvalovský. Top-down neural model for formulae. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Byg5QhR5FQ.

[4] Xavier Glorot, Ankit Anand, Eser Aygün, Shibl Mourad, Pushmeet Kohli, and Doina Precup. Learning representations of logical formulae using graph neural networks. In *Workshop on Graph Representation Learning at 33rd Neural Information Processing Systems (NeurIPS 2019)*, 2019. URL https://grlearning.github.io/papers/58.pdf.

[5] Jan Jakubuv, Karel Chvalovský, Miroslav Olšák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA Anonymous: Symbol-independent inference guiding machine (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020. ISBN 978-3-030-51053-4. doi:10.1007/978-3-030-51054-1_29.

[6] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013. ISBN 978-3-642-39798-1. doi:10.1007/978-3-642-39799-8_1.

[7] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017. doi:10.29007/8mwc.

[8] Miroslav Olšák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 1395–1402. IOS Press, 2020. ISBN 978-1-64368-100-9. doi:10.3233/FAIA200244.

[9] Michael Rawson and Giles Reger. Directed graph networks for logical reasoning (extended abstract). In Pascal Fontaine, Konstantin Korovin, Ilias S. Kotsireas, Philipp Rümmer,

and Sophie Tourret, editors, *Joint Proceedings of the 7th Workshop on Practical Aspects of Automated Reasoning (PAAR) and the 5th Satisfiability Checking and Symbolic Computation Workshop (SC-Square) Workshop, 2020 co-located with the 10th International Joint Conference on Automated Reasoning (IJCAR 2020), Paris, France, June-July, 2020 (Virtual)*, volume 2752 of *CEUR Workshop Proceedings*, pages 109–119. CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2752/paper8.pdf.

[10] Stephan Schulz. *E 2.4 User Manual*, 2020. URL https://easychair.org/publications/preprint/8dss.

[11] Stephan Schulz and Martin Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2016. ISBN 978-3-319-40228-4. doi:10.1007/978-3-319-40229-1_23.

[12] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 59(4), December 2017. ISSN 0168-7433. doi:10.1007/s10817-017-9407-7.

[13] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. doi:10.1016/j.aiopen.2021.01.001.