Integrating graph neural networks into cvc5

Jelle Piepenbrock[1,2], Mikoláš Janota[2], Josef Urban[2], and Jan Jakubův[2,3]

[1] Radboud University, Nijmegen, the Netherlands
jelle.piepenbrock@ru.nl
[2] Czech Technical University in Prague, Prague, Czech Republic
firstname.lastname@cvut.cz
[3] University of Innsbruck, Innsbruck, Austria

Introduction: In satisfiability modulo theories (SMT) solvers, the ground reasoning is handled by highly optimized theory-specific solvers. For example, linear integer arithmetic or boolean vector calculations can be handled by specific routines that exploit domain knowledge.

The non-ground reasoning can be handled via quantifier instantiation. In existing SMT solvers, such as cvc5 [1], there are already several strategies implemented to choose how to instantiate quantified expressions (QEs). For example, there is enumerative instantiation, in which the solver prefers tuples of terms that contain terms created earlier. There is also E-matching, which uses a specific pattern matching algorithm to choose terms to instantiate with. Another possibility is to make use of the propositional model, like in model-based quantifier instantiation. As the space of possible instantiations is difficult to navigate, a logical step is to use machine learning techniques to learn a heuristic that will determine which instantiations are preferred by the solver. Here we modify the enumerative instantiation with an ML predictor.

Graph Neural Network in cvc5: While many different neural network methods can be used to guide automated theorem provers, a natural choice, based on the graph representation that cvc5 uses for the proof state, is the class of predictors called *graph neural networks* (GNNs) [6]. On a high level, GNNs represent each node in a graph with a vector of floating point numbers, and update these vectors using the vector representations of neighbouring nodes in the graph. By using optimization procedures, the GNN 'learns' to aggregate and update the node representations in such a way that at the end of several iterations of this neighbourhood-based updating procedure (usually called *message passing*), the node representations contain useful information to predict some relevant quantity. In our setting, these relevant quantities are: (i) scores for each quantified expression that represent whether this expression should be instantiated and (ii) scores for each pair of variables and terms that represent whether this particular variable should be instantiated with a particular term.

We implement a custom GNN using the C++ frontend of PyTorch [5]. ML-guided ATPs often use a separate GPU server [2, 3], to which multiple prover processes send their requests for advice. Here, we are however interested in a tight integration within cvc5. The network is trained on e-matching proof traces on first-order CNF problems extracted from the Mizar Mathematical Library and then used to predict instantiations in the enumerative mode. This is because the e-matching mode is stronger on the Mizar data, and can thus gather more training data, but the enumeration mode is simpler to modify and instrument with a GNN. In Table 1, we show the development and holdout set performance of the ML-guided cvc5, along with various control experiments. One experiment is a *dry run* to measure slowdown effects: we compute the neural network predictions but ignore them and use the standard enumeration age-based heuristics instead. There are two ways to use the QE predictions: probabilistic sampling or a threshold. We observe that the performance of the enumeration mode was improved by up to 72% (336/195 = 1.723) when we use the Threshold (1 inst) variant of our network.

	Development	Holdout
Bcvc5 — only e-matching	1096	1107
Bcvc5 — only enumeration	183	195
MLcvc5 — dry run	119	120
MLcvc5 — model (QSampling)	288	300
MLcvc5 — model (Threshold - 1x10-5, 1 inst)	324	336
MLcvc5 — model (Threshold - 1x10-5, 10 inst)	410	407

Table 1: MML1147: Number of problems solved by 10s runs. On both the development and the holdout sets the GNN-guided enumeration mode outperforms the unguided enumeration mode. Both the development and holdout set contain 2896 problems.

These results taken together indicate that the network has learned a useful strategy from the e-matching generated training data, which it can apply in the ML enumeration mode.

The data in Figure 1 indicates that some difference in performance is due to the difference in the raw number of instantiations done. As we are already incurring the cost of computing the GNN advice, it might be the case that instantiation with multiple high-scoring tuples per round, instead of only 1 per QE as the original enumeration does, is a better use of the GNN advice. To test this, we ran a version of the ML mode that performs up to 10 instantiations per QE per round (see Table 1). This led to 407 solved holdout problems (again in 10s real time). This is a 109% increase compared to the unmodified enumeration mode ($407/195 = 2.09$).

Conclusion & Future Work: While e-matching largely dominates on first-order logic problems extracted from the Mizar Mathematical Library, on problems from the SMTLIB database, the enumeration procedure is much more competitive [4], and can even outperform e-matching on certain types of benchmarks. In principle, we can extend the current method to SMT problems, aside from the fact that the logging procedure that extracts training data from cvc5 runs needs to be modified.

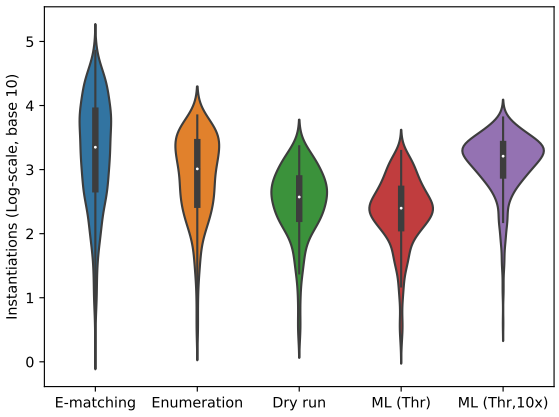

Figure 1: Violin plots of the number of instantiations performed in successful runs for Bcvc5 e-matching, Bcvc5 enumeration, dry run, the ML strategy with threshold 1e-5, and the ML strategy with 10x as many instantiations per quantifier per round. The white dots indicate the medians. The respective medians are 2235, 1026, 373.5, 250 and 1620.

References

[1] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. "cvc5: A Versatile and Industrial-Strength SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. DOI: 10.1007/978-3-030-99524-9_24. URL: https://doi.org/10.1007/978-3-030-99524-9_24.

[2] Karel Chvalovský, Konstantin Korovin, Jelle Piepenbrock, and Josef Urban. "Guiding an Instantiation Prover with Graph Neural Networks". In: *LPAR*. Vol. 94. EPiC Series in Computing. EasyChair, 2023, pp. 112–123.

[3] Zarathustra Amadeus Goertzel, Jan Jakubův, Cezary Kaliszyk, Miroslav Olsák, Jelle Piepenbrock, and Josef Urban. "The Isabelle ENIGMA". In: *ITP*. Vol. 237. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 16:1–16:21.

[4] Mikoláš Janota, Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. "Fair and adventurous enumeration of quantifier instantiations". In: *2021 Formal Methods in Computer Aided Design (FMCAD)*. IEEE. 2021, pp. 256–260.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.