Guiding an Instantiation Prover with Graph Neural Networks*

Karel Chvalovský[1], Konstantin Korovin[2], Jelle Piepenbrock[13], and Josef Urban[1]

[1] Czech Technical University in Prague
[2] University of Manchester, Manchester, U.K.
[3] Radboud University Nijmegen

Introduction One of the most challenging tasks in the area of AI/TP systems is equipping fast state-of-the-art automated theorem provers (ATPs) with *efficient internal guidance* of their calculi based on learning from many previous proof-search decisions. While related AI/TP tasks such as learning-based *premise selection* [1], *tactical guidance* [4], and *neural conjecturing* [17] can use relatively slow and expensive ML methods that are called only rarely on a single problem, internal guidance requires efficient ML methods and their nontrivial integration with the fast ATPs. Several advances in internal guidance were recently made for connection-based [9, 14, 18] and resolution/superposition-based ATPs [7, 15, 6]. In this work, we develop a strong internal ML guidance for iProver [10] that is based on an instantiation calculus.

iProver iProver [10] is an ATP for quantified first-order logic. At the core of iProver is an instantiation calculus, Inst-Gen [3, 11], which can be combined with resolution and superposition calculi [2]. The Inst-Gen calculus is based on propositional reasoning to deal with propositional abstractions of first-order clauses and model-guided incremental instantiations using unification to generate new first-order instances. Although this approach often works well in practice, one of the major bottlenecks is the number of generated first-order instances, and only a few of them are usually needed in the final proof. Hence we investigate here how machine learning can be used to select clauses most likely to be used in a proof.

We have developed an interactive mode for iProver, described in detail in our repository.[1] In this mode, iProver communicates with an *external agent (EA)* via TCP/IP sockets. The external agent can provide proof search guidance by either assigning scores to clauses that are used for prioritising them for the next inferences or explicitly selecting the given clause.

GNN Guidance We have developed[2] an external agent that uses a graph neural network (GNN) to score clauses generated in the Inst-Gen calculus. In particular, we use a PyTorch [13] implementation of the name-independent GNN architecture developed originally for connection provers [12], but also used in the recent E/ENIGMA systems [5]. The main function is similar to E/ENIGMA, where a Python GPU server [6] is used to reduce the overhead of repeated model loading. However, in E/ENIGMA, the server is *stateless*. Here, we instead build a richer *stateful* server that keeps track of the given clauses and conjectures, and is capable of generating the context used to score the requests.

Experimental Setting The evaluation is performed on a large benchmark of 57 880 problems[3] originating from the Mizar Mathematical Library (MML) [8] and exported to first-order

*The full paper was recently accepted to LPAR 2023.
[1] https://gitlab.com/korovin/iprover/-/blob/master/README-interactive.md
[2] https://gitlab.ciirc.cvut.cz/chvalkar/iprover-gnn-server
[3] http://grid01.ciirc.cvut.cz/~mptp/1147/MPTP2/problems_small_consist.tar.gz

logic by MPTP [16]. The Mizar problems are split[4] (in a 90-5-5% ratio) into *training*, *development*, and *holdout* sets. Since we are interested in internal ATP guidance, we use problems with premises limited to those used in the human-written Mizar proofs (*bushy problems*).

The training data are collected from previous successful runs. In the previous ENIGMA systems, each proof (*classical* data) gives a training example where given clauses are considered useful or useless depending on whether they ended up in the proof. However, the size of such a training example corresponds to the final size of the set of all given clauses, not to the sizes of actual score requests performed during a proof search. To address this problem, we newly experiment with the actual score requests (*dynamic* data) from the successful proof searches, where all clauses not used in the proof (including passive clauses) are considered useless.

We run iProver in its instantiation mode, but a combination with superposition and resolution is also possible. Score requests are made in batches to improve performance.

In the setting where the scores provided by the EA are used for clause selection, we use two modes: *solo* and *coop*. In the solo mode, there is just one priority queue ordered by the scores provided by the EA. In the coop mode, we combine EA-provided scores with the human-programmed priority queues in an equal ratio.

Results We run iProver with the human-programmed priority queues for clause selection to collect the initial training examples. In this mode, without being slowed down by the server (non-interactive mode), iProver solves 502 theorems out of the 2896 theorems in the development set in 15 seconds. However, to extract the initial training data, we need to run iProver in the interactive mode (ignoring meaningless scores); it solves 482 (no GNN called) and 451 theorems (with a randomly initialized GNN) in the development set, respectively. Therefore, there is some overhead from the communication and GNN calculations, but its impact is manageable.

We train three models, one with classic data and two with dynamic data (taking randomly 4 or 10 queries from the successful runs, respectively). Each model is run in either solo or coop mode. We used two best-performing models to obtain further training data for the next iteration and repeated this process twice. Interestingly, dynamic data performed significantly better than static data, and the solo mode dominated the coop mode in later iterations.

iProver, using guidance from our best-performing model, solves 1094 problems in the development set and 1093 in the holdout set in 15 seconds. Hence solving more than twice as many problems as iProver in the non-interactive mode using the same time limit. Moreover, it solves a very similar fraction of problems on the training set. Similar results also hold for other models. The training procedure seems to be quite robust to overfitting on the training data and to generalize well. This could be due to several aspects: (i) different runs lead to different sets of given clauses, (ii) only a limited number of the dynamic samples is seen during the training per problem (4 or 10), and (iii) the contexts are randomly sampled from the available given clauses. Interestingly, when we evaluate just the accuracy of the trained GNN model, its performance on the train, development, and holdout sets slightly differs. This difference is probably not large enough to cause significant differences in the ultimate ATP performance.

As an additional measure of generalization, we also evaluated the trained system on the 13 370 theorems in 242 articles that were added in a newer version of MML (1382) and thus never seen during training. More than half of these problems contain new terminology. Whereas iProver in the non-interactive mode solves 1662 theorems, iProver guided by our best model solves 3657 theorems. Hence the improvement is similar to the results on the dataset that we used for our training.

[4] http://grid01.ciirc.cvut.cz/~mptp/Mizar_eval_final_split

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.

[2] André Duarte and Konstantin Korovin. Implementing superposition in iProver (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 388–397. Springer, 2020.

[3] Harald Ganzinger and Konstantin Korovin. New directions in instantiation-based theorem proving. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 55–64. IEEE Computer Society, 2003.

[4] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Tactictoe: Learning to prove with tactics. *J. Autom. Reason.*, 65(2):257–286, 2021.

[5] Zarathustra A Goertzel, Jan Jakubův, Cezary Kaliszyk, Miroslav Olšák, Jelle Piepenbrock, and Josef Urban. The isabelle enigma. In *13th International Conference on Interactive Theorem Proving*, 2022.

[6] Zarathustra Amadeus Goertzel, Karel Chvalovský, Jan Jakubuv, Miroslav Olsák, and Josef Urban. Fast and slow Enigmas and parental guidance. In *FroCoS*, volume 12941 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2021.

[7] Jan Jakubuv and Josef Urban. Hammering Mizar by learning clause guidance. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, volume 141 of *LIPIcs*, pages 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[8] Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reasoning*, 55(3):245–256, 2015.

[9] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8836–8847, 2018.

[10] Konstantin Korovin. iProver - an instantiation-based theorem prover for first-order logic (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 292–298. Springer, 2008.

[11] Konstantin Korovin. Inst-Gen - A modular approach to instantiation-based automated reasoning. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics - Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 239–270. Springer, 2013.

[12] Miroslav Olšák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In *ECAI 2020*, pages 1395–1402. IOS Press, 2020.

[13] 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. *Advances in neural information processing systems*, 32, 2019.

[14] Michael Rawson and Giles Reger. lazycop: Lazy paramodulation meets neurally guided search. In *TABLEAUX*, volume 12842 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2021.

[15] Martin Suda. Improving enigma-style clause selection while learning from history. In *CADE*, volume 12699 of *Lecture Notes in Computer Science*, pages 543–561. Springer, 2021.

[16] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*,

37(1-2):21–43, 2006.

[17] Josef Urban and Jan Jakubuv. First neural conjecturing datasets and experiments. In Christoph Benzmüller and Bruce R. Miller, editors, *Intelligent Computer Mathematics - 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, volume 12236 of *Lecture Notes in Computer Science*, pages 315–323. Springer, 2020.

[18] Zsolt Zombori, Josef Urban, and Miroslav Olsák. The role of entropy in guiding a connection prover. In *TABLEAUX*, volume 12842 of *Lecture Notes in Computer Science*, pages 218–235. Springer, 2021.