Descending to Complementarity*

Martin Suda

Czech Technical University in Prague, Czech Republic

Motivation

A proving *strategy* is a specific configuration of an automatic theorem prover (ATP) used to attack a problem. It is broadly recognized that there is no universally good strategy and that, instead, different strategies work well on different problems. At the same time, the task of predicting in advance, given a particular problem, which strategy will perform best on it, seems very hard. For these reasons, ATPs typically run—in their most powerful mode of operation—a whole sequence of time-limited strategies called a *schedule*, where the individual strategies are selected to complement each other. This idea of *time slicing* has been first implemented in the ATP Gandalf [10, 11] and has been since successfully adopted by most modern ATPs (cf. the auto-schedule mode in E [6] or the CASC mode in Vampire [3]). A plausible explanation for why *strategy scheduling* is so effective in practice, is the observation that if a problem can be solved at all, it can often be solved by some strategy very quickly [5].

In recent years, we have seen several promising approaches improving the performance of ATPs with the help of machine learning, in particular artificial neural networks. Most notable are systems targeting for the improvement *clause selection*, a crucial heuristic choice point in a saturation-based ATP. Examples of such systems include ENIGMA-anonymous [1], where clause selection is guided with the help of a graph neural network [4], and Deepire [7], which enhances Vampire's clause selection capabilities by an advice from a recursive neural network constructed over clause derivation history. While the improvement by these ML-integrations is very encouraging if not downright impressive, their common trait is that they only strive to improve a single base prover strategy, thus ignoring the power of strategy scheduling.

The aim of this work is to enhance systems such as ENIGMA or Deepire with the ability to simultaneously train a full host of guiding heuristics, i.e., strategies, which automatically adapt to the training problems and become complementary in aspects relevant to solving them. In the method we propose, the individual strategies arise as local specialists on automatically clustered problems that are most efficiently solved by the respective variant of the learned heuristic (such as clause selection). These specialists are determined as small "tweaks" of the universally optimal, generalist strategy that would be obtained normally.

Method

The main idea is to assign a small learnable embedding vector v_p to each input problem p and to *condition* the network being trained N by v_p when learning from examples or experiences coming from p. The conditioning could mean providing v_p as a separate input, but other schemes seem possible and will be experimented with as part of this work. During training with gradient descent the embedding v_p will "travel" from its initial value (e.g., $v_p = \vec{0}$ for every p) to encode a (mild) specialization—for its corresponding problem p—of the overall general guiding heuristic. At the same time, the training will thus implicitly cluster the input problems according to which variation of the general heuristic is most suitable for solving them.

*Supported by the project RICAIP no. 857306 under the EU-H2020 programme and the Czech Science Foundation project 20-06390Y.

In more detail, let us imagine we are training a network N from experiences collected while running our prover over a set of problems P. Our training examples will be of the form $(input, target, p)$ where $p \in P$ is a problem on which the prover was run to obtain the example. In standard training with gradient descent, our network would be determined by a vector of parameters θ and we would, for each example, compute the gradient

$$\nabla_\theta Loss(N_\theta(input), target)$$

to update these parameters, where $N_\theta(input)$ is the network's prediction for *input* (which would—during inference time—be used for the prover guidance).

With conditioning, we additionally maintain a set of problem embeddings $\{v_p\}_{p \in P}$ to be treated as additional trainable network parameters. Our gradient formula becomes

$$\nabla_{\theta, v_p} Loss(N_{\theta, v_p}(input), target)$$

when training on an example obtained while running the prover on p.

The additional degrees of freedom should allow the network to achieve an overall smaller loss. Thus we expect N_{θ,v_p} (i.e., the specialist) to be more effective at solving problem p (and similar ones) than the plain N_θ (i.e., the generalist). However, it is important that the dimension of the embedding $|v_p|$ is small compared to number of parameters $|\theta|$, as we still want the essence of the learned guidance to reside in θ and each v_p to be only a small "tweak" of this essence.[1]

At the same time, our overall aim is for the space spanned by the embeddings $\{v_p\}_{p \in P}$ to exactly represent the space of all reasonable strategies representable by our architecture. The method guarantees that "relevant complementarity" arises for free: the loss decreases whenever v_p specializes the guidance in aspects relevant for solving p better than the generalist would. There is still a risk, however, that not every point in the spanned space will correspond to a reasonable heuristic for the ATP (imagine a point lying too far from any final learned embeddings v_p). We will take inspiration from the *variational autoencoder* models [2], which face similar difficulties, and adapt their solution of adding an explicit clustering factor into the loss function, if needed. Then, for deployment, equipped with a whole space of heuristics, we will be able to sample new strategies on demand, taking advantage of the previously solved clustered problem to navigate the corresponding space.

Proof of Concept Implementation

In the talk, we will report on experiments with the described idea in the context of reinforcement learning for clauses selection in the ATP Vampire. This is a continuation of our work motivated in an AITP presentation last year [8]. We train a feed-forward neural network to classify clauses based on a small set of standard features such as age, weight, the number of variable occurrences, or a distance to the goal. This setup is deliberately close to the human-designed heuristics (such as age/weight alternation) aiming to answer the question: can these be re-learned just from the prover experience? And deliberately simple to be competitive in real time evaluation.

Another distinguishing features of our approach is that we do not attempt to model the prover's state for influencing the decisions. This is again in line with the state-of-the-art human-designed heuristics and allows us to store clauses, as usual, in a queue data structure (once evaluated by the network) for an efficient "extract-min" retrieval. However, we treat both the prover and the guiding agent stochastically [9] which brings some extra challenges.

Without aspiring to explain all the relevant details here (thus mainly for the aesthetic enjoyment of the kind reader) a small teaser of our results is presented in Figure 1.

[1] At the opposite extreme, we would learn a separate network N_p for every problem p, which would most

References

[1] J. Jakubuv, K. Chvalovský, M. Olsák, B. Piotrowski, M. Suda, and J. Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, vol. 12167 of *Lecture Notes in Computer Science*, pp. 448–463. Springer, 2020.

[2] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[3] L. Kovács and A. Voronkov. First-order theorem proving and vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, vol. 8044 of *Lecture Notes in Computer Science*, pp. 1–35. Springer, 2013.

[4] M. Olsák, C. Kaliszyk, and J. Urban. Property invariant embedding for automated reasoning. In *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)*, vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1395–1402. IOS Press, 2020.

[5] G. Reger. Boldly going where no prover has gone before. In *Proceedings of the Second International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements, ARCADE@CADE 2019, Natal, Brazil, August 26, 2019*, vol. 311 of *EPTCS*, pp. 37–41, 2019.

[6] S. Schulz, S. Cruanes, and P. Vukmirović. Faster, higher, stronger: E 2.3. In *Proc. of the 27th CADE, Natal, Brasil*, number 11716 in LNAI, pp. 495–507. Springer, 2019.

[7] M. Suda. Vampire with a brain is a good ITP hammer. In *Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, vol. 12941 of *Lecture Notes in Computer Science*, pp. 192–209. Springer, 2021.

[8] M. Suda. Elements of reinforcement learning in saturation-based theorem proving. In *7th Conference on Artificial Intelligence and Theorem Proving AITP 2022 – proceedings*, 2022. http://aitp-conference.org/2022/abstract/AITP_2022_paper_11.pdf.

[9] M. Suda. Vampire getting noisy: Will random bits help conquer chaos? (system description). In *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, vol. 13385 of *LNCS*, pp. 659–667. Springer, 2022.

[10] T. Tammet. Towards efficient subsumption. In *Automated Deduction - CADE-15, 15th International Conference on Automated Deduction, Lindau, Germany, July 5-10, 1998, Proceedings*, vol. 1421 of *Lecture Notes in Computer Science*, pp. 427–441. Springer, 1998.

[11] T. Tammet. Towards efficient subsumption – full version: http://www.cs.cmu.edu/~fp/courses/atp/cmuonly/T98.pdf, 1998. Accessed on May, 2023.

---

likely not generalize to previously unsolved problems.

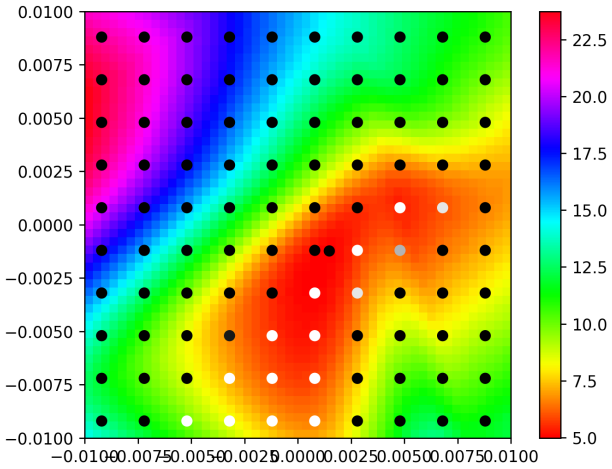

Figure 1: A heat map showing guiding model's loss as a function of a two-dimensional problem "tweak" embedding v. The loss was computed for a particular trace collected from solving the TPTP problem `GRA002+1`. The loss achieves a minimum of around 5.0 in the central red patch. In the regular grid overlay of the small circles, the actual success rate of Vampire guided with a particular tweak is shown in grayscale (black means "never solved", pure white is "solved in 10 out of 10 randomly seeded experimental runs"). The black circle that is not part of the grid represents the actual current "tweak" v_p for the mentioned problem. The plot demonstrates a rare situation when a loss-minimizing version of the guidance does not lead to a success, while other versions of the guidance do. This points to an imperfect match between the loss function and the loss-minimizing network's influence on the prover performance.