Learning Strategy Design: First Lessons

Martin Suda[1]*and Sarah Winkler[2]†

[1] Czech Technical University in Prague, Prague, Czech Republic
martin.suda@gmail.com
[2] Università degli Studi di Verona, Verona, Italy
sarahmaria.winkler@univr.it

Abstract

Automatic theorem provers typically offer a wide variety of parameters to control their search strategy. However, the strategy range is vast, and a suitable strategy for a given input problem hard to predict.

Here we sketch an experiment analyzing a large data set, obtained by running all 801 of Vampire's CASC mode 2018 strategies on first-order problems in TPTP: (1) We build random forest regression models predicting strategy performance, using different feature sets to investigate feature importance. (2) Each strategy is defined by a set of parameter values. We investigate the correlation between problem features and successful parameter values to work towards learning how to construct suitable strategies for a given problem. (3) We analyze correlations between problem features and the success of tools run in CASC.

1 Strategy Prediction

Data set. Vampire [6] was run for 60 s on all 17 574 FOL problems in TPTP library [9] (version 7.2.0) using all the 801 strategies used in CASC-27 (a total running time of ∼16 years on a single core).

Problem features. We use the problem properties specified in the TPTP files (e.g., number of axioms, terms, variables, etc) and a property set determined by Vampire (e.g., has extensionality, linear integer or group problem). Moreover we experimented with the three TPTP *pseudo-features*: domain, rating, and source,[1] where "source" refers to a token combining the author and year of the TPTP submission, like 'Sla93'. In addition, three hand-crafted features were used which estimate the number of unifiable positive and negative literals and the number of terms matching and unifying with (non-variable) equation sides. In total, we obtained 98 features in this way.[2]

Regression models. For each of the 801 strategies (set $\mathcal{S}$), we built a random forest regressor to predict the runtime on a problem using our feature set [8]. Hyperparameters were determined by a grid search. We built rating-balanced test and training sets (ratio 1:4), and trained all 801 regressors on the latter. (If a strategy did not solve a problem, a *timeout penalty* of 300 s was assigned.) In the test phase, for every problem in the test set, the strategy with the lowest predicted run time was recommended, and we counted how many problems can be solved by the recommended strategy. The lessons we learned from that include the following:

*Supported by the ERC Consolidator grant AI4REASON no. 649043 under the EU-H2020 programme and the Czech Science Foundataion project 20-06390Y.

†Supported by FWF project T789.

[1] One can question whether these are "legitimate" features as they are not obtained from the problem itself viewed as a logical formula.

[2] For the feature list and all further details see http://profs.scienze.univr.it/winkler/learn_strat/.

- *Tell me the source, I tell you the strategy.* When predicting a strategy in $\mathcal{S}$ from a *single* feature, the source works best: 2342 of 3515 test problems can be solved (2180 from the number of terms, 2241 from the domain, 2166 from the rating). From all features, successful predictions can be made for 2583 problems, without TPTP features 2548.
- *Interaction matters.* All three handcrafted features about unifiable and matching terms are in the top 10 of the most important features. In total, they contribute 11.6%. Other top 10 features are numbers of terms (6.2%), variables (4.3%), atoms (3.8%), connectives (3.6%), functions (3.5%), unit clauses (3%), and constants (3%) (without TPTP features). If included, rating is *the* most important feature (23%) and source is in the top 10, too.
- *Regression quality ≠ prediction power.* The *coefficient of determination* (r^2-value) is a common measure for the amount of variance explained by a regression model. When using all features it amounts to $r^2 = 0.71$, for source only $r^2 = 0.28$, for rating only $r^2 = 0.41$. Obviously, the rating can correctly predict many timeouts, but as the numbers of solved problems above show, not so many suitable solving strategies.

2 Correlation

In Section 1 we tried to predict a strategy from the fixed set $\mathcal{S}$. Next we investigate correlations between problem features and strategy components (i.e., Vampire options). To that end, we clustered problems according to features and compared the probability that a problem from some cluster C can be solved by a strategy with option o set to a particular value v to the probability that (1) an arbitrary strategy solves a problem from C, and (2) a strategy with $o = v$ solves a problem on average. For example, on EPR problems, age-weight ratio (-awr; used for controlling clause selection) values of 1:50, 1:64, 1:128 are 11.0%, 8.6%, 9.6% better than strategies are on EPR on average, and 15.0%, 16.8%, 18.0% better than these option values usually are. As another example, for the sources 'Sla93', 'WM89', 'Bau99', 'Sta09', problems are 60% more likely to be solved by a strategy with the finite model builder (-sa fmb; which replaces a saturation algorithm). We add two more general observations (for more examples and complete data see the website):

- *The strongest correlations appear with sources.* Even for sources which occur in at least 20 problems, we found 389 correlations where problems from a particular source are solved at least 30% more likely with a certain option.
- *Correlations identify fragile options.* For options like nwc[3] and awr, often a certain range is beneficial (as for EPR above), but others are *fragile*, i.e. only one value works.
- *EPR and UEQ show correlations for many option values.*

We also correlated feature properties with the success rate of different CASC tools, using TPTP2T data to check which tool is (a) the most appropriate for a cluster, and (b) more powerful on a cluster than on TPTP in general. For (a), though Vampire is almost always the best choice, we found some clusters where this is not the case. For instance, in the presence of reals CVC4 1.7 is superior, with list axioms Leo-III 1.3 and Isabelle 2016 prevail, and for problems with source 'Hoe08' or 'Sta08', versions of E solve most problems. For (b), we discovered many cases of "overperformance" of a tool on a certain problem cluster. For instance on EPR, iProver, Z3, and Zipperposition win by overperformance, but Vampire solves more problems.

[3]The non-goal-weight coefficient: it penalises clauses not derived from the conjecture by artificially increasing their weight (by multiplying it by a given coefficient) before it is used for clause selection.

Related Work. Given the numerous parameters offered by state-of-the-art theorem provers and the hence vast number of search strategies, automatic tuning by machine learning techniques is a natural approach. Early work in this direction was done for the equational theorem prover Discount [2, 3] using a syntactic feature set and a nearest-neighbor strategy. A similar approach was also pursued for E using features related to symbol counts, evaluated both a priori and after some proof steps [1]. Similar features, together with the TPTP properties, were exploited by the strategy tuning framework MaLeS [7]. The strategy design tools BliStr and BliStrTune [5] predict strategies for E and were evaluated on the Mizar Mathematical Library. Related work for iProver was also presented at AITP 2019 [4].

Conclusion. In next steps, we want to repeat our experiments with a more extensive strategy set, take the solving time for correlations into account, and play with dimensionality reduction. In the future, we plan to work on predicting good *schedules* [10] for a given problem, which is a potentially very rewarding endeavour but has received relatively little attention so far.

References

[1] J. P. Bridge, S. B. Holden, and L. C. Paulson. Machine learning for first-order theorem proving - learning to select a good heuristic. *Journal of Automated Reasoning*, 53(2):141–172, 2014.

[2] M. Fuchs. Automatic Selection of Search-Guiding Heuristics. In D. D. II, editor, *Proc. of the 10th FLAIRS, Daytona Beach*, pages 1–5. Florida AI Research Society, 1997.

[3] M. Fuchs. *Learning Search Heuristics for Automated Deduction.* Number 34 in Forschungsergebnisse zur Informatik. Verlag Dr. Kovač, 1997. Accepted as a Ph.D. Thesis at the Fachbereich Informatik, Universität Kaiserslautern.

[4] E. K. Holden and K. Korovin. SMAC and XGBoost your theorem prover. In *Proc. 4th Conference on Artificial Intelligence and Theorem Proving (AITP 2019)*, pages 93–95, 2019.

[5] J. Jakubuv and J. Urban. BliStrTune: hierarchical invention of theorem proving strategies. In Y. Bertot and V. Vafeiadis, editors, *Proc. 6th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2017)*, pages 43–52. ACM, 2017.

[6] L. Kovács and A. Voronkov. First-order theorem proving and Vampire. In N. Sharygina and H. Veith, editors, *Proc. 25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

[7] D. Kühlwein and J. Urban. MaLeS: A framework for automatic tuning of automated theorem provers. *Journal of Automated Reasoning*, 55(2):91–116, 2015.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[10] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.