# Representing First-order Problems for Heuristic Selection

Edvard K. Holden and Konstantin Korovin

The University of Manchester, U.K.

**Abstract**

Machine learning (ML) has been applied to Automated Theorem Provers (ATPs) with much success in recent years. However, representing first-order problems effectively as feature vectors remains a major challenge. The performance of an ML approach is bounded by how well the features represent the problems and relates to the task at hand. In this paper, we investigate the effectiveness of different problem representations for heuristic selection embeddings.

## Heterogeneous Heuristic Selection and Optimisation

Heuristics are crucial for the success of ATPs, but finding good heuristics is challenging due to their vast parameter space. Additionally, problems are heterogeneous which means that different heuristics are required to solve different problems. Although there have been substantial work on automatically discovering heuristics for first-order reasoning [9, 12, 15, 17], they do not consider the heterogeneous nature of first-order problems.

To tackle this challenge, we developed a system for *heterogeneous heuristic optimisation and scheduling* using machine learning (HOS-ML) [4], illustrated in Figure 1. The key idea behind HOS-ML is to dynamically partition heterogeneous problem sets into homogeneous problem clusters and optimise heuristics for each cluster separately using Bayesian hyper-parameter optimisation. While there is no obvious way of grouping problems into homogeneous clusters, in [4] we proposed to compute clusters through a combination of heuristics evaluation clustering and problem embedding.

HOS-ML consists of three phases. The first phase discovers new heuristics by interleaving Bayesian hyper-parameter optimisation for discovering promising heuristics and dynamic re-clustering into homogeneous problem clusters. The problems are iteratively re-clustered into increasingly finer-grained clusters using heuristics evaluation feature vectors, which are extended as we discover new heuristics. In the second phase, we use constraint programming to construct effective schedules for each homogeneous cluster based on the discovered heuristics. The final phase deploys cluster schedules on unseen problems.

A core component of HOS-ML, used in the final phase, is the *heuristics embedding model*, as illustrated by the node "Embed" in Figure 1. The heuristics embedding model maps unseen problems into heuristics evaluation vectors which in turn are used to assign unseen problems into clusters and the corresponding cluster schedules. The heuristics embedding model relies on problem representation using feature vectors. Such problem representations are useful in many applications but present a major challenge. In the following we experimented with different problem representations in the context of HOS-ML.

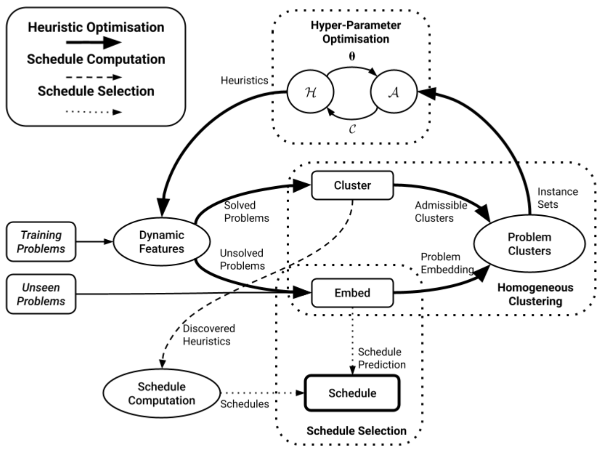

Figure 1: HOS-ML: heuristic optimisation and selection for heterogeneous problems.

## Problem Representation

A significant challenge is that many ML algorithms and models operate on numerical vectors known as feature vectors, while ATPs deal with sets of formulas. Therefore, applying ML on the formula level requires a feature vector representation of the formulas. However, there is no natural way for mapping tree-structured first-order formulas into one-dimensional numerical vectors.

This impediment becomes even more apparent at the problem-level for tasks such as heuristic selection. Problems consists of sets of formulas and the feature vector must be able to create a representation of this set. These challenges has typically been addressed by using more attainable feature sets at the cost of information loss about the problem's structure [1, 3, 5, 6, 14], or complex graph-based embeddings [8, 13] which can be difficult to train. The features have to contain some information related to the task at hand. For embedding evaluation properties, this means that the features should reveal some information about the behavioural properties of the problem, such that we can predict how a heuristic is going to perform on it. In our experiments we considered three different feature types for this task:

- **Syntactic Features:** These include syntactic properties such as the number equational, Horn, EPR, ground formulas, etc. We used 14 of such features. Such features create a good representation of the problem encoding but do not always reflect the algorithmic properties of formulas.

- **Solver State Features:** Syntactic features do not always reflect solver performance, additionally, the original structure of a problem does not always correspond to the internal representation of the problem in the ATP. To overcome this issues we consider solver state features. These features consist of 155 solver statistics on the solvers' key function calls during a run of the prover including a range of simplification counts and timings. These features are computed by attempting a problem with a single heuristic for a low-timelimit and extracting the solver statistics after termination. The advantage of solver state features is that they directly represent performance of different components of the solver on the problem, in contrast to the syntactic features.

- **Abstract Features:** Symbol based representations can be effective [7, 11] but have a drawback of being sensitive to symbol renaming. One approach to this problem is to use embeddings based on sophisticated graph neural networks [8, 13]. In this paper we investigate a simpler approach using *signature abstraction* by collapsing all signature symbols of the same type into an abstract symbol of this type. Abstract symbols are shared across all problems resulting in common features even when problems have different signatures. Signature abstraction preserve variable dependencies and fragmentic structure of the formulas, e.g., of being EPR, Horn, ground, monadic, etc. After creating an abstraction of the problem on the term, literal and clause level we represent the abstracted problem as a bag-of-words. In our experiments we only used the abstract features computed from the conjectures of the problems.

## Problem Embedding

Let $H$ be a set of heuristics, and $\mathbf{s_p}$ the feature vector of $p$. The admissible embedding model $\mathcal{E}$ learns the mapping between $\mathbf{s_p}$ and the evaluation vector $\mathbf{e_p}$, which encodes the evaluation of the heuristics in $H$ on $p$. We use multi-label classification to construct the embedding model $\mathcal{E}$. We separate the multi-label classification task into $|H|$ binary classification tasks, where a separate binary classification model $\mathcal{M}_\theta$ is trained for each heuristic $\theta$ in $H$. The final embedding model is defined as $\mathcal{E}(\mathbf{s_p}) = (\mathcal{M}_{\theta_1}(\mathbf{s_p}), \cdots, \mathcal{M}_{\theta_{|H|}}(\mathbf{s_p}))$, and can be used to predict the evaluation vector of any given problem. In this paper, we predict which of the heuristics can solve a problem within a given time limit. We have experimented with different binary classification models such as neural networks, SVMs and tree-based model. In our experiments, the random forest model XGBoost [2] yielded the best performance.

| | Syntactic | Solver | Abstract | All |
|---|---|---|---|---|
| F1-Score | 0.76 | 0.80 | 0.72 | 0.81 |
| Geometric Accuracy | 0.68 | 0.75 | 0.65 | 0.76 |
| Hamming Loss | 0.71 | 0.76 | 0.67 | 0.76 |

Table 1: The embedding performance of different problem representations on testing problems.

## Evaluation

In this experiment we evaluated the quality of the three types of problem representations in the context of HOS-ML: using syntactic, solver state and abstract features. The experiment consists of training the heuristic embedding model for each feature set and evaluating it on a set of unseen test problems. To obtain the experiment problems, we randomly sampled 4000 FOF and CNF problems from the TPTP library (v7.4.0) [16]. Next, we extracted the features of each problem and removed all problems that were solved during feature extraction or did not parse within a 1-second time limit.

To obtain the evaluation vectors we evaluated seventeen iProver [10] heuristics on the experiment problems with a 300 second time limit. Further, we removed the problems that were unsolved by all the heuristics as well as problems with all solutions below five seconds. This results in a problem set consisting of challenging yet solvable problems. Finally, we divided the remaining problems into training and testing sets with a 70-30 split.

We trained heuristic embedding models using XGBoost and different problem representations as described in the previous section. From the results shown in Table 1, we observe that heuristic embedding models can predict heuristic evaluation vectors with high accuracy. The solver state features outperform the two other feature types. We also observe that the performance is slightly increased by combining all three feature sets.

The heuristic embedding model was integrated in our HOS-ML implementation and we are currently evaluating HOS-ML on the full TPTP.

## References

[1] James P. Bridge, Sean B. Holden, and Lawrence C. Paulson. Machine learning for first-order theorem proving. *Journal of Automated Reasoning*, 53(2):141–172, Aug 2014.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[3] Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. *CoRR*, abs/1903.03182, 2019.

[4] Edvard K. Holden and Konstantin Korovin. Heterogeneous heuristic optimisation and scheduling for first-order theorem proving. CICM 2021. Submitted.

[5] Edvard K. Holden and Konstantin Korovin. Experiments with selection of theorem proving heuristics. In *Proc. Automated Reasoning Workshop 2019 Bridging the Gap between Theory and Practice*, 2019.

[6] Edvard K. Holden and Konstantin Korovin. SMAC and XGBoost your theorem prover. In *Proc. 4th Conference on Artificial Intelligence and Theorem Proving*, 2019.

[7] Geoffrey Irving, Christian Szegedy, Alexander A. Alemi, Niklas Eén, François Chollet, and Josef Urban. Deepmath - deep sequence models for premise selection. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243, 2016.

[8] 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.

[9] Jan Jakubuv and Josef Urban. Blistrtune: hierarchical invention of theorem proving strategies. In Yves Bertot and Viktor Vafeiadis, editors, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France*, pages 43–52. ACM, 2017.

[10] Konstantin Korovin. iProver - an instantiation-based theorem prover for first-order logic (system description). In *IJCAR 2008. Proceedings*, pages 292–298, 2008.

[11] A. S. Kucik and K. Korovin. Premise selection with neural networks and distributed representation of features. *ArXiv e-prints, Abs/1807.10268*, July 2018.

[12] Daniel Kühlwein, Stephan Schulz, and Josef Urban. E-males 1.1. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, 2013. Proceedings*, volume 7898 of *LNCS*, pages 407–413. Springer, 2013.

[13] Miroslav Olsák, 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.

[14] Michael Rawson and Giles Reger. Dynamic strategy priority: Empower the strong and abandon the weak. In Boris Konev, Josef Urban, and Philipp Rümmer, editors, *6th Workshop on Practical Aspects of Automated Reasoning (PAAR)*, number 2162 in CEUR Workshop Proceedings, pages 58–71, Aachen, 2018.

[15] Simon Schäfer and Stephan Schulz. Breeding theorem proving heuristics with genetic algorithms. In Georg Gottlob, Geoff Sutcliffe, and Andrei Voronkov, editors, *GCAI 2015. Global Conference on Artificial Intelligence*, volume 36 of *EPiC Series in Computing*, pages 263–274. EasyChair, 2015.

[16] 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.

[17] Josef Urban. Blistr: The blind strategymaker. In Georg Gottlob, Geoff Sutcliffe, and Andrei Voronkov, editors, *Global Conference on Artificial Intelligence, GCAI 2015, Tbilisi, Georgia, 2015*, volume 36 of *EPiC Series in Computing*, pages 312–319. EasyChair, 2015.