Project Proposal: Model-Based Optimization of Strategy Schedules*

Filip Bártek and Martin Suda

Czech Technical University in Prague

1 Introduction of the main task

Automated theorem provers (ATPs) such as Vampire [4] are parameterized by a large number of options that control the proof search heuristics. We call each valid combination of option values of a given prover a strategy.

Given a prover A, a problem $p \in \mathcal{P}$, and a strategy $s \in \mathcal{S}$, the outcome of running A on p using s often varies greatly depending on s. For example, an execution of Vampire with the option `--saturation_algorithm fmb` (finite model building) [4] searches for a model of the problem and thus is only suitable for satisfiable problems, while the saturation algorithm `discount` searches for a refutation by iterative inference and is especially suitable for unsatisfiable problems. On a problem set that contains both satisfiable and unsatisfiable problems, strategies with the saturation algorithms `fmb` and `discount` are complementary.

Strategy scheduling [6] is a common way to utilize the complementarity of various strategies: Given a problem, the prover runs a sequence of independent time-limited solution attempts, each with a different strategy. A question naturally follows: Given a prover, a problem, and a runtime budget, how can we find a strategy schedule that optimizes the expected performance of the prover on the problem?

Several successful attempts have been explored to answer this and other closely related questions. Vampire offers a so-called CASC mode [4], which chooses a schedule based on the syntactic features of the input problem. MaLeS [5] first initializes a set of useful strategies and then dynamically constructs a schedule of these strategies for an input problem. Given a constraint satisfaction problem (CSP), CPHYDRA [2] schedules eight solvers optimally with respect to a model trained in advance. Hydra [8] constructs a portfolio of strategies using an arbitrary algorithm configurator (for example, Sequential Model-based Algorithm Configuration (SMAC) [3]) and a portfolio builder (for example, SATzilla [9]).

In this proposed project, we intend to design and evaluate a method for data-parsimonious construction of a strategy schedule: Given a prover and a distribution of problems, the method should propose a schedule with a high expected probability of solving a problem from the input distribution, while keeping the number of evaluations of the prover during the training small.

2 Solution outline

We propose to search for a good schedule by Sequential Model-Based Optimization (SMBO) [3], that is, by iterating between fitting a regression model of runtime for a given problem and strategy, and gathering additional training data.

*This work is supported by the Czech Science Foundation project no. 20-06390Y (JUNIOR grant), and the Grant Agency of the Czech Technical University in Prague, grant no. SGS20/215/OHK3/3T/37.

The training data consists of triples of the form (p, s, t), where $p \in \mathcal{P}$ is a problem, $s \in \mathcal{S}$ is a strategy, and $t \in \mathbb{R}$ is a PAR10[1] [1] score of solving problem p with strategy s. Each problem is represented by a vector of syntactic features, and each strategy is defined as a vector of numeric and categorical values. Thus, it is possible to fit an empirical performance model $\hat{\pi}(T|P, S)$ that predicts the PAR10 score T (as a random variable) for problem P and strategy S. We will model the score by a random forest or a Gaussian process, which will allow us to capture the uncertainty of the model. We denote the ground truth score distribution by $\pi(T|P, S)$.

A schedule $f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ allocates a time slice to each strategy. For practical reasons, we assume that f allocates a nonzero time slice to only a finite number of strategies. When the prover runs with a strategy s on problem p, the probability of failure to solve the problem in time limit $f(s)$ is $\pi(T > f(s)|P = p, S = s)$. The probability that the schedule f fails to solve problem p is $\prod_{s \in \mathcal{S}, f(s)>0} \pi(T > f(s)|P = p, S = s)$. An optimum schedule $f^*_{\pi,B}$ minimizes the expected number of unsolved problems $C = \sum_{p \in \mathcal{P}} \prod_{s \in \mathcal{S}, f(s)>0} \pi(T > f(s)|P = p, S = s)$ under the runtime budget constraint $\sum_{s \in \mathcal{S}, f(s)>0} f(s) \leq B$.

To sample a new data point, we choose a problem p and a strategy s in a way that combines exploration and exploitation in the context of strategy scheduling using our current score model $\hat{\pi}$. The choice of the sampling scheme is the central research question of this project.

Once the training has finished, it is necessary to extract a schedule from the performance model $\hat{\pi}$. We plan to construct a complete schedule by splitting the time budget into slices, and iteratively assigning each slice either to a new strategy found by local search, or to one of the strategies that have already been assigned at least one slice, minimizing the expected number of unsolved problems C greedily.

To assess the practical utility of the proposed system and especially the schedule-directed sampling scheme, the system will be implemented and empirically compared with a combination of the greedy schedule constructor Hydra [8] and the general-purpose algorithm configurator SMAC [3], which uses a sampling scheme oriented at finding a single best strategy.

3 Discussion

The space of valid strategies is vast, which makes the search for useful strategies difficult compared to the task of algorithm selection. The training problem set is typically large and contains groups of similar problems. Generalizing across strategies and problems becomes crucial to modeling the runtime in a data-efficient manner. Care must be taken not to overfit to the training problem set.

The trained model may overfit to a particular first-order logic representation of the input problem, while many equivalent representations of the problem exist. E.g., equivalent representations that differ only in the order of clauses, the order of literals in clauses, the orientation of equalities, or the names of the symbols. Probabilistic modeling may allow us to capture this ambiguity in a rigorous fashion.

The choice of the schedule quality criterion depends on the intended use. As an alternative to the expected number of problems solved within the runtime budget, the expected time to solve an arbitrary problem could be optimized.

The task of budgeted strategy scheduling is parameterized by the runtime budget B. A question of potential interest is: Can a schedule construction scheme be trained without reference to a particular runtime budget value?

[1] For a solution attempt, the penalized average runtime with a penalty factor of 10 (PAR10) is the actual runtime in case the solution is found within the time limit, or 10 times the time limit in case the attempt times out. Alternatively, survival analysis [7] may allow a more rigorous handling of timeouts.

References

[1] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, aug 2016.

[2] Derek Bridge, Eoin O'Mahony, and Barry O'Sullivan. Case-based reasoning for autonomous constraint solving. In *Autonomous Search*, volume 9783642214349, pages 73–95. Springer-Verlag Berlin Heidelberg, oct 2012.

[3] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6683 LNCS, pages 507–523. Springer, Berlin, Heidelberg, 2011.

[4] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

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

[6] Tanel Tammet. Towards efficient subsumption. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1421, pages 427–441. Springer Verlag, 1998.

[7] Alexander Tornede, Marcel Wever, Stefan Werner, Felix Mohr, and Eyke Hüllermeier. Run2survive: a decision-theoretic approach to algorithm selection based on survival analysis. In *Asian Conference on Machine Learning*, pages 737–752. PMLR, 2020.

[8] Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 210–216. AAAI Press, 2010.

[9] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, jul 2008.