Characterizing Algorithm Features in Automated Algorithm Selection

Stephen, Austin
University of Wyoming
astepehn@uwyo.edu

Abstract

When evaluating sets of computationally challenging problems with sets of complimentary algorithms the meta-algorithmic technique of automated algorithm selection has shown major success. Traditionally, these automated algorithm selection models are trained on the problem instance feature values and performance data from algorithm runs. Our work shows training these automated algorithm selection models on the feature values of the algorithms in addition to problem instance features improves overall performance. This allows the evaluation of sets of problem instances in less time and with less memory by choosing the most optimal algorithm more frequently.

SAT and Automated Algorithm Selection

To study the effectiveness of automated algorithm selection our work used the problem instances and solvers from SAT competitions as a proxy for computationally difficult tasks with complimentary approaches to evaluation. A Boolean satisfiability formula, called a SAT instance, is a set of variables related by propositional logic. A solver is an algorithm that determines if the given SAT instance has variable assignments that result in it being true/satisfiable or false/unsatisfiable. The solver that preforms the best across an entire set of instances is referred to as the single best solver. SAT instances are considered NP-complete, as a result, formally evaluating them can be computationally expensive when they are of appreciable size. Therefore, SAT solvers are designed to leverage heuristics or shortcuts for evaluating instances. Efficient algorithms often evaluate a given instance in seconds while inefficient algorithms can take hours or never arrive at a solution. Developing these solvers that evaluate SAT instances is a regularly revisited challenge and competitions benchmark state-of-the-art SAT solvers. Different solvers exhibit different performance characteristics depending on how features of the instance relate to the heuristics the algorithm uses to evaluate it. Often different solvers performance is complementary, meaning some solvers perform well on one family of instances while others are more efficient on different families. Therefore, meta-algorithmic methods like automated algorithm selection use machine learning on portfolios of solvers attempting to employ the best solver on a given instance. If the optimal solver in a portfolio was selected for every instance this is referred to as the virtual best solver. The performance of an algorithm selection model can be determined by comparing how much of the gap between the single and virtual best solver it bridges. Automated algorithm selection has shown performance improvements in the area of SAT, with early successes including "SATzillla" (Xu et al 2008). Automated algorithm selection historically uses machine learning algorithms trained on partitioned sets of results from individual solvers evaluation of instances and feature values of each instance. The features of an instance can include information about the construction of the Boolean formula like the number of "for all" quantifiers in the instance. These are extracted from the set of instances using a feature extraction algorithm run on every instance.

Training Algorithm Selection Models on Algorithm Features

Historically, algorithm selection models are trained solely on the feature values of the instances and treat the algorithm as a black box. This project examines the performance improvements conferred by training these models on the algorithm feature values on top of the instance features. Prior to this work there was not an automated algorithm selection model that takes advantage of this information. Extracting the algorithm features is done using a single model for the entire portfolio and only requires the source code of the solvers. Feature extraction from the source code means the

inclusion of algorithm features does not require new information on the scenario. Furthermore, feature extraction is done at runtime to avoid harming the ability of the model to generalize as this was the traditional motivation for excluding algorithm features. On the ASlib benchmark library this approach generated improvements up to 32 percent in misclassification penalty.

Related Work

Automated algorithm selection has shown it can generate improvements on sets of problem instances without the addition of new algorithms. Furthermore, it adds utility to algorithms that may not universally improve on solving problems in a given space but improve on a subset of this space. This is important because developing entirely new algorithms is a difficult task and seldom offers universal improvement without tradeoffs (Kotthoff 2014). Automated algorithm selection has been extended to QBF, my focus area. Using a combination of a local search solver and portfolio of heuristics showed improvements in evaluating a set of QBF instances. (Samulowitz and Memisevic 2007). QBF is a viable proxy for this research because of the difficulty associated with evaluating a QBF instance as they are PSPACE-hard (Palo, et al 2016). Also, QBF has seen a recent resurgence of importance in the SAT community. In 2016 there was the first QBFEVAL competition in 5 years that pulled out a new set of state-of-the-art solvers. These competitions often set the standards for research as was the case for QBF and SAT. (Paulina and Seidl 2019).

My Contribution

Completed Progress

The high-level view of my work on this project is building a dataset based on the QBF solver competitions. In constructing this dataset, we hope to extend the performance improvements experienced from training on algorithm features in SAT to QBF. A quantified Boolean formula (QBF) is a subarea of SAT where every variable in a problem instance has an existential quantifier attached. QBF instances must be evaluated by solvers specifically built for QBF, and as a result, have separate competitions. My work for this project began with identifying the solvers used in historical QBF competitions and searching for places the authors made them publicly available. Once I found a solver in a public repository, I built and compiled it. This required a containerization application built for high-performance computing called Singularity because many state-of-the-art solvers are five or more years old and are not compatible with modern compilers or architectures. Therefore, I used Singularity images to replicate the environment the solvers originally competed in and contained the correct version of the solver's dependencies. Once the solvers were built, I aggregated the instances from the repository at qbflib.org in both input formats used for the competition, QDIMACS, and QCIR. Using a conversion script, I made a universal set of each input type converting all instances into the format they are not native to. This enabled the comparison of solvers from different years that use different input formats on the same set of instances. This step is partially why it is not possible to conduct analysis from the competition results alone. After the solvers and instances are aggregated, compiled, and converted, I ran every solver on every instance to provide the baseline data needed for algorithm selection. This used the local high-performance computing cluster and jobs were submitted using batch scripts and the SLURM schedular. Using 5 sovler. The memory and time parameters used for these jobs was configured to match those of the competition the solvers were originally entered in and are all equal. Having finished collecting the data about algorithm performance on instances, I then use a feature extraction algorithm to extract the feature values of the instances. This information is used to train the automated algorithm selection model. Lastly, I converted the performance data to the ASlib format, which is the standardized format used by researchers to compare their algorithm selection scenarios and is necessary to build the automated algorithm selection model.

Working Progress

As of the submission of this extended abstract I am still in the progress of finishing the following work, as a result, it is subject to minor methodological changes as unforeseen events arise. Currently, I am building the algorithm selection models using the performance results from the solver runs and the instance feature values. I am building the algorithm selection models in R using the llama package for algorithm selection. In total, I will construct four algorithm selection models using the llama package: classification, classification pairs, clustering, and regression pairs. These four models offer a diverse baseline making for stronger comparisons to the models trained on algorithm features. After the models are completed, I will utilize the work from the other researchers working on this project to make models trained on algorithm features as well. Then I will conduct analysis of the QBF scenario using empirical performance models and performance difference models. This analysis is modeled after the analysis conducted on other scenarios and will enable me to add my results to the body of evidence showing the viability of using algorithm features in algorithm selection.

References

Kotthoff L. 2014. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Magazine.* 35(3) 48-60.

Marin P, Narizzano, M., Pulina L., Tacchella A., Giunchiglia E. 2016. Twelve Years of QBF Evaluations: QSAT Is PSPACE-Hard and It Shows. Special issue of the 22nd RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion. doi.org/10.3233/FI-2016-1445.

Paulina, L; Seidl, M. 2019. The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17) *Artificial Intelligence* 274: 224-248. https://doi.org/10.1016/j.artint.2019.04.002

Samulowitz H.; Memisevic R. 2007 Learning to solve QBF. In the Proceedings of Association for the Advancement of Artificial Intelligence 22nd National Conference on Artificial Intelligence. 255-260.

Xu, L., Hutter F., Hoos H., Leyton-Brown. 2008 SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal Of Artificial Intelligence Research* 32: 565-606. https://doi.org/10.1613/jair.2490