Using Mixture of Experts Method in Combining Search-Guiding Heuristics for Theorem Proving

C. Acar Erkek[1] and Tunga Güngör[2]

Boğaziçi University
Bebek, 34342 Istanbul, Turkey
[1] acarerkek@gmail.com
[2] gungort@boun.edu.tr

Abstract. The main challenge of automated theorem proving is to find a way to shorten the search process. Therefore using a good heuristic method is essential. Instead of constructing a heuristic from scratch, we propose to use the mixture of experts learning to combine the existing heuristics to construct a heuristic from similar problems. The results show that the combined heuristic is better than each individual heuristic used in combination.

1 Introduction

The resolution principle reduces proof procedures into a series of simpler, unintelligent operations, which computers perform very fast. Although the resolution principle is useful and fast, it has no predefined clause choosing mechanism and it expands the search space quickly. So, it is essential to use a heuristic method to narrow (or direct) the search path.

In automated theorem proving (ATP) systems, usually static evaluation functions are used as heuristics. These functions are dependent on the features of the clauses (e.g. the number of symbols in a clause). Usually problems have different characteristics and require different approaches. We do not have "the best" heuristic which is successful for each problem. Machine learning methods can be used for inventing good heuristics, improving existing ones, adapting methods for the given problem [1], or choosing a suitable heuristic from a given set [2].

Mixture of experts (MOE) is a variant of artificial neural networks, and it can be used to combine multiple learning or non-learning experts. Its main purpose is to learn the regions where each expert is successful. In this paper, we propose a novel method for automated theorem proving, based on the MOE approach to combine different heuristics, and to construct a new heuristic. This heuristic is shown to be more successful than each individual heuristic used independently.

2 Previous Work

A neural network can be used to learn the search-guiding heuristics [3]. The training data of the neural network are the proofs of non-heuristic version. The

steps that contribute to the proof are taken as positive training data. Branches that do not contribute but are close to the positive training data are taken as negative examples.

The similarity between problem definitions is used in [4]. Solutions of solved (similar) problems are used to configure the parameters of the heuristic, to be used in the current problem. Machine learning is used to suggest a sequence of heuristics according to their similarity with the current problem [2]. Numeric features of the problem descriptions (axioms and the conclusion) are used to define the similarity. Also, two different heuristics can be learned and then combined. The success of the combination is better than both of the heuristics in some of the tests [5].

Although converting a clause into numeric representation causes some loss of information, numeric representations are usually used since they are suitable for inexact knowledge, we can define similarity and distance concepts easily, and there are powerful learning methods with numeric representations [1]. Numeric features of clauses are used to convert them into numeric representations. Number of literals (in a clause), number of distinct predicates, number of variables, number of functions and term depth of the clauses are examples.

3 Proposed Method

The given-clause algorithm is a popular and efficient algorithm used in ATP systems [6]. A good heuristic function is essential for this algorithm since after some time, the number of inferences explodes.

MOE can be seen as a general architecture for combining multiple experts, where the experts may not be linear or learning and the gating may not be linear [7],[8]. The idea is to achieve success rates better than each individual expert. MOEs are trained with the back-propagation algorithm. We prefer the cooperative learning model, since it is shown that cooperative model is more accurate than the competitive model [7].

We use clause heuristics as the experts of the system. The flexibility of MOE allows us to use different types of heuristics together (non-learning heuristics and learning heuristics). Different experts may give output in different scales, which affects the learning process negatively in early stages. We propose to filter the outputs of experts with perceptrons (which are trained separately for each expert), therefore we do not use the outputs of experts directly, but posterior probabilities, which are calculated by perceptron, are used in mixture of experts. This method also ensures that outputs of experts are in $[0, 1]$ interval.

In applying machine learning, our training data will be the output of the system, which indicates that we will use the proof steps of previous problems to solve new problems. Initially, since there is no training, we must use the outputs of problems solved with conventional heuristics. We choose clauses that contribute to the solution as positive examples, that do not contribute to the solution as negative examples. a proof has much more negative examples than positive examples. It is better to take negative examples which are close to

positive examples [1]. We only include negative examples which are two steps away from positive examples in the proof tree.

The positive and negative examples are converted into their numeric representations. This numeric data are used to train the MOE network. We train the network until the coefficients are stable. In our examples, all training sessions are very fast (takes less than 1 s.), so compared to the proof sessions, the total time of the training sessions is negligible. Some of the clause features used in numerical representation are number of literals, predicates, constants, functions, variables, maximum nesting of the clause and maximum weight of literals.

The problem definition of the new problem is compared with the previously solved problems and the knowledge of the most similar problem is applied to the new problem. This concept, which is called instance-based learning, is successfully used in [2]. To determine the similarity, each problem is converted into numerical representations. Then, similarity is calculated as the euclidean distance between these feature vectors. Some of the features used in the application are term depth of the axioms, number of distinct predicates and function arities. In the future, a mechanism should be implemented for dealing with the problems that do not have applicable knowledge in their close neighborhood.

The MOE is initialized with the coefficients taken from the most similar problem. And the output of this MOE network is used as the heuristic function in the given-clause algorithm.

4 Experiments and Discussion

In our experiments, we implemented the proposed method on top of the Otter ATP system [6], by modifying the clause selection mechanism of Otter to use the mixture of experts. The other mechanisms of Otter were kept the same so that we can isolate the effect of the clause selection mechanism in the results.

The experiments were done on an Intel Pentium 4 1.7 Ghz Ubuntu Linux computer. In all of the tests, a moderate time limit (3 min.) was given to the prover to prevent running indefinitely if it does not find a solution. We used the TPTP (Thousands of Problems for Theorem Provers) library in the tests [9]. We used problems without equality, which are defined in clause normal form.

In the experiments, we combined three simple heuristics. For comparison, we used two hypothetical heuristics: For a given problem, one of these hypothetical heuristics acts as the best (H_{best}), the other acts as the worst (H_{worst}) among other heuristics (H_1, H_2, H_3). And our learned heuristic is H_{comb}. Other experts can be added to the system (learning or non-learning). The system will easily be integrated with experts which use back-propagation without any modification.

The results of experiments for the FLD domain are given in Figure 1. Results show that in 35% of the problems, H_{comb} is at least as fast as the hypothetical heuristic H_{best}. So, we can conclude that the system has gained abilities beyond the combined heuristics for these problems. Also we should consider that constructing the perfect H_{best} heuristics is impossible and all heuristics will be subject to the problems of similarity and numerical representations.

For some problems (10%), we see that, our proposed system is worse than combined heuristics. There are two possibilites for these negative results: the loss of information due to numerical representations and the similarity approach we used. An analysis of negative results should help us to improve our similarity approach. An alternative for numerical representations is symbolic representation approach, which can be combined with the current work in the future with a slight modification in the gating structure [10].

Both H_{comb} and H_{best} succeeded	82	51%
Both H_{comb} and H_{best} failed	72	45%
H_{comb} succeeded but H_{best} failed	5	3%
H_{comb} failed but H_{best} succeeded	2	1%
H_{comb} is faster than H_{best}	23	14%
H_{comb} is as fast as H_{best}	35	21%
H_{comb} is faster than H_{worst} but slower than H_{best}	10	6%
H_{comb} is slower than H_{worst}	14	9%
Total number of problems	161	100%

Fig. 1. Number of solved problems from FLD domain

References

1. J. Denzinger, M. Fuchs, C. Goller, and S. Schulz.: Learning from Previous Proof Experience: A Survey. Technical Report AR-99-4, Fakultät für Informatik der Technischen Universität München, 1999.
2. M. Fuchs.: Automatic Selection of Search-Guiding Heuristics. Proc. of FLAIRS, pages 1–5, 1997.
3. C. Suttner and W. Ertel.: Automatic Acquisition of Search Guiding Heuristics. Proc. of International Conference on Automated Deduction, pages 470–484, 1990.
4. M. Fuchs and M. Fuchs.: Applying Case-Based Reasoning to Automated Deduction. Proc. of International Conference on Case-Based Reasoning, pages 23–32, 1997.
5. M. Fuchs.: Experiments in the Heuristic Use of Past Proof Experience. Proc. of CADE-13, New Brunswick, LNAI, 1104:523–537, 1996.
6. W. McCune: OTTER 3.3 Reference Manual. Argonne National Laboratory, Technical Memorandum No.263, 2003.
7. E. Alpaydin.: Introduction To Machine Learning. MIT, 2004.
8. S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards.: Artificial Intelligence: A Modern Approach. Prentice Hall, NJ, 1995.
9. G. Sutcliffe.: The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. Journal of Automated Reasoning, 43(4):337–362, 2009.
10. C. Goller.: A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems. PhD Dissertation, Technical University of Munich, 1999.

This article was processed using the LaTeX macro package with LLNCS style