Learning Heuristic Functions Faster by Using Predicted Solution Costs

Levi H. S. Lelis
Computing Science Dept.
University of Alberta
Edmonton, Canada
(santanad@cs.ualberta.ca)

Shahab Jabbari Arfaee
Computing Science Dept.
University of Alberta
Edmonton, AB, Canada
(jabbaria@cs.ualberta.ca)

Sandra Zilles
Computer Science Dept.
University of Regina
Regina, Canada
(zilles@cs.uregina.ca)

Robert C. Holte
Computing Science Dept.
University of Alberta
Edmonton, Canada
(holte@cs.ualberta.ca)

Abstract

Jabbari Arfaee, Zilles, and Holte presented the bootstrap learning system, a system that learns strong heuristic functions for state-space problems. They showed that IDA* with a bootstrap heuristic is able to quickly find near-optimal solutions in several problem domains. However, the process the bootstrap method uses to learn heuristic functions is time-consuming: it is on the order of days. In this paper we present a learning system that uses an approximation method instead of an exact one to generate the training set required to learn heuristics. We showed recently that solution costs can often be quickly and accurately predicted without having to actually find a solution. In this paper we apply this idea to speedup the process of learning heuristics. In contrast with other learning approaches that use search algorithms to solve problem instances to generate the training set, our system uses a solution cost predictor. We reduce the time required to learn strong heuristics from days to minutes on the domains tested.

Introduction

Heuristic search algorithms such as IDA* (Korf 1985) use a heuristic function to guide the search. The use of machine learning to learn a heuristic function recently became popular, see for instance (Ernandes and Gori 2004), and (Jabbari Arfaee, Zilles, and Holte 2011). One of the main challenges faced by such learning systems is to collect training instances; in order to learn heuristics one must know the solution cost of a sufficiently large number of problem instances. Jabbari Arfaee et al. (2011) presented a learning system, Bootstrap, that generates training data through bootstrapping. Bootstrap tries to solve problem instances with a (possibly weak) initial heuristic h_0 within a time limit. The instances that the method manages to solve form a training set that is used to learn another, stronger heuristic h_1. The process is then repeated with h_1 replacing h_0, hoping that some of the instances not solved in the previous iteration will be solved and a new training set will be obtained. IDA* finds near-optimal solutions for state-space problems once a bootstrap heuristic is learned. However, the bootstrap learning process is time-consuming – it could take days to finish.

The bootstrap process is time-consuming for two reasons. First, since there is no easy way of knowing in advance whether an instance will be easy or hard to solve, Bootstrap spends a substantial amount of time trying to solve instances to be used for training. Second, Bootstrap learns from the easy instances first. Therefore, it can take several iterations until a heuristic is created that is able to solve hard problem instances quickly.

In this paper we propose `BiSS-h`, a novel approach to learning heuristics with the goal of reducing learning time. Instead of using search to solve problem instances to generate a training set, we use a prediction method to estimate the solution cost of the training instances (Lelis, Stern, and Jabbari Arfaee 2011). The solution cost predictor we use to generate the training set is `BiSS` (Lelis et al. 2012).[1]

By using predictions instead of search to generate the training set, our system `BiSS-h` learns heuristics substantially faster than Bootstrap. This is because `BiSS` is able to produce a prediction of the optimal solution cost for any problem instance, no matter whether it is easy or hard to solve. Thus, our method does not have to spend time attempting and failing to solve problem instances. Furthermore, in contrast with the bootstrap method that often requires a large number of instances to be solved before it learns from hard problem instances, our method is able to learn from hard problem instances even with a small training set. This is because `BiSS-h` does not require a strong heuristic to generate training data from hard instances.

The contributions of this paper are empirical: we show that it is possible to quickly learn strong heuristics from data generated by a solution cost predictor.

Experimental Results

We ran experiments on the (5x5) 24-puzzle, the 35 pancake puzzle, and Rubik's Cube. We compare `BiSS-h` solely to Bootstrap since (1) we evaluate its learning time and (2) in terms of search performance, IDA* using a bootstrap heuristic outperforms standard suboptimal search algorithms on the domains tested (Jabbari Arfaee, Zilles, and Holte 2011).

The learning algorithm we use in our experiments is the

[1]Note that although `BiSS` produces accurate estimates of the optimal solution cost on the domains tested, it is too slow to be used directly as a heuristic function.

same used with bootstrap (Jabbari Arfaee, Zilles, and Holte 2011). Also, for bootstrap and BiSS-h we always used exactly the same set of features. For BiSS we used the same parameters used in its original paper (Lelis et al. 2012). As we are interested in learning effective heuristics quickly, we use with BiSS-h the smallest training sets reported by Jabbari Arfaee et al. (2011) (500 training instances). Smaller training sets result in shorter learning time. We use a combination of easy and hard instances in our training sets: For the 24-puzzle and the 35 pancake puzzle, 400 out of the 500 instances were generated with random walks from the goal. The length of each random walk was chosen randomly between 1 and 50 moves. The 100 remaining training instances were generated randomly. For Rubik's Cube, we generated easy and hard training instances solely with random walks from the goal. The length of each random walk for Rubik's Cube was chosen randomly between 1 and 80 moves.

We evaluate Bootstrap and BiSS-h primarily on the basis of the time required to learn a heuristic. To ensure that the heuristics learned are of roughly the same quality, we also measure the average runtime, and average suboptimality of the solutions. We compute the suboptimality for one problem instance as follows. We divide the cost found by the optimal cost and then subtract one from the result of the division. The average suboptimality represents the percentage average in which the solutions found by an algorithm overestimates the optimal solution cost. We present units of time in our table of results with the letters d, h, m, and s indicating days, hours, minutes, and seconds, respectively.

Algorithm	Learning Time	Solving Time	Subopt.
24-puzzle			
BST	11h 43m	64s	5.7
BiSS	17m	77s	5.0
35 Pancake Puzzle			
BST	1d 11h	48s	5.5
BiSS	33m	33s	5.0
Rubik's Cube			
BST	2d	8,253s	4.0
BiSS	53m	2,148s	11.4

Table 1: Learning heuristic functions with BiSS.

Table 1 presents the results. BiSS-h can learn equally strong heuristics in substantially less time compared to Bootstrap. An important fact to note from the data in Table 1 is that this is the first time, to the best of our knowledge, that near-optimal solutions have been found for Rubik's Cube in substantially less time than optimal solutions — the fastest known average optimal solving time for random instances of the Rubik's Cube is due to Zahavi et al. (2008): 12 hours, 16 minutes and 41 seconds on average.

The Selection of Training Instances Matters

Experiments on the 20 blocks world show that the strategy for collecting the training instances matters. IDA* with the heuristic our method learns using the same strategy as in the other experiments (i.e., using a mix of easy and hard training instances) solved only 10 out of the 50 test instances with a time limit of one hour per instance. BiSS-h clearly failed to learn a strong heuristic in this case considering that IDA* solves the same instances in about 30 seconds on average when using a bootstrap heuristic. We then ran our system again but, instead of using the training set consisting of random walk instances and truly random instances, we used the instances used by Bootstrap on its last iteration, including all the instances on a path to the goal found by Bootstrap. With these instances our system learned a much stronger heuristic than Bootstrap. This shows that the training instances play a crucial role on the quality of the heuristic learned.

Conclusions

In this paper we presented BiSS-h, a learning system that uses a solution cost predictor instead of a search algorithm to generate the training set required to learn heuristics. BiSS-h is able to learn effective heuristics much faster than Bootstrap — a learning method that uses a search algorithm to generate the training set. Our system reduces the time required by Bootstrap to learn heuristics from days to minutes.

Our results on the blocks world suggest that by carefully selecting training instances one could learn much stronger heuristics. To the best of our knowledge, our learning system is the only one that gives the flexibility of choosing which instances are used for training. Previous systems are restricted to the instances that are solvable by a search method to form the training set. As BiSS can be used to predict the solution cost of virtually any problem instance, our system opens up the possibility of applying active learning techniques (Settles 2010) to learning heuristics — a subject of future work.

Acknowledgements

This work was supported by the LCD at the University of Regina, AITF, AICML, and NSERC.

References

Ernandes, M., and Gori, M. 2004. Likely-admissible and sub-symbolic heuristics. In *ECAI*, 613–617.

Jabbari Arfaee, S.; Zilles, S.; and Holte, R. C. 2011. Learning heuristic functions for large state spaces. *Artificial Intelligence* 175(16-17):2075–2098.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27:97–109.

Lelis, L.; Stern, R.; Felner, A.; Zilles, S.; and Holte, R. C. 2012. Predicting optimal solution cost with bidirectional stratified sampling. In *ICAPS*.

Lelis, L.; Stern, R.; and Jabbari Arfaee, S. 2011. Predicting solution costs with conditional probabilities. In *SoCS*.

Settles, B. 2010. Active learning literature survey. Technical Report 1648, Computer Sciences. University of Wisconsin–Madison.

Zahavi, U.; Felner, A.; Holte, R. C.; and Schaeffer, J. 2008. Duality in permutation state spaces and the dual search algorithm. *Artificial Intelligence* 172(4–5):514–540.