CS365: Artificial Intelligence
Project Proposal
Learning Heuristic Functions for Large State Spaces

Vinit Kataria(vinitk@iitk.ac.in) , N.V.Subba Rao(subbarao@iitk.ac.in)
Guide: Dr. Amitabha Mukerjee

Introduction & Motivation:

The pressing need of real-world problems for implementable and practically feasible solutions has increased the significance of good heuristic functions in modern times. It has been realized that many important planning applications are so challenging that obtaining an optimal solution is not practical. Hence, in most of the cases, the main approach to such problems is to try to obtain solutions in reasonable time and at the cost of optimality, if needed, but keeping in mind that these solutions are not far from optimal. Many of the current real-world problems can be transformed into path-finding problems. To reduce the size of search space, it is critical to use effective heuristics.

A popular approach to creating heuristics for a state space is abstraction. One of the limitations of this approach is that it is usually memory-intensive. Moreover, since combinatorial problems grow exponentially, for the problems to be faced, with the computers of foreseeable future, even the best heuristics created by similar approaches will be too weak to enable arbitrary instances to be solved reasonably quickly. One of the approaches to the automatic creation of heuristics that gets over these limitations is to apply machine learning to learn heuristic functions.

Many studies have been carried out on the popular sliding-tile puzzles, which turn out to be complex enough to highlight the most relevant problems. The above idea of automatic creation of heuristics has been applied with great success to the 15-puzzle and other state spaces of similar size (see Samadi, Felner, and Schaeffer [2] and Ernandes and Gori [3]), but could not be applied to larger spaces, like the 24-puzzle, because of the excessive time it would take to create a sufficiently large training set containing a sufficiently broad range of possible distances to goal.

Ernandes and Gori [3] proposed a different way of extending the machine learning approach to scale to arbitrarily large problems(but never implemented it) called the “bootstrap learning of heurisitic functions”(or bootstrapping).

Objective:

In this project, we plan to use machine learning to create effective heuristics for IDA* search algorithm to solve single instances of sliding-tile puzzle (one of the problems having large state space which is complex enough to highlight the most relevant problems) in reasonable time at the cost of optimality. We aim to test this method on the 15 puzzle and, if time permits, on the 24 puzzle. The bootstrapping procedure that we plan to implement is supposed to solve single instances of problem quickly but compromising with the optimality within reasonable bounds by using an interleaving method to carry on the learning and solving processes in parallel.

<u>Methodology:</u>

The key concept of our approach is to generate stronger heuristics starting from weaker ones using a bootstrapping procedure. The main algorithms and methods to be used in the procedure are:

- Bootstrap algorithm : An iterative procedure that starts from a weak heuristic and a set of unsolved instances(which not necessarily can be solved using this weak heuristic) and develops stronger heuristics successively from the training sets generated during the procedure.
- IDA* algorithm: The search algorithm used to solve problem instances in this process.
- Random walk algorithm : In case the current heuristic(during the bootstrap procedure) cannot solve at-least some threshold number of bootstrap instances, this algorithm is used to generate instances starting from goal state to create instances which are easy enough to be solved by the current heuristic.
- Learning algorithm : Artificial neural networks would be used to learn heuristics using training set generated during the bootstrapping procedure. The admissibility of the learned heuristic is relaxed in probabilistic sense.
- Interleaving procedure : In this method, we use a fixed ratio of the time allocated to solving and the time allocated to learning which would help solving single problem instances in practically reasonable time.

<u>Related Work:</u>

Similar work has been carried out in the last decade in this area by Ernandes & Gori in 2004 in which admissibility of the heuristics was relaxed in probabilistic sense but could not be applied to larger state spaces because of excessive time required to create sufficiently large quality training set, then in 2008 Samadi, Felner, Schaeffer used multiple heuristics but reverted to abstraction approach inheriting its limitations. Later in 2010, Jabbari, Zilles, Holte used similar bootstrapping procedure to solve problems in larger problem domains like 24 puzzle, rubik's cube, etc.

<u>References:</u>

- Shahab Jabbari Arfaee, Sandra Zilles, Robert C. Holte. *Learning Heuristic Functions for Large State Spaces.* In Elsevier,175:pages 2075–2098, 2011.
- Jonathan Schaeffer, Ariel Felner, Mehdi Samadi. *Learning from multiple heuristics.* In proceedings of 23rd AAAI Conference on Artificial Intelligence(2008), pages 357-362, 2008.
- Marco Ernandes and Marco Gori. *Likely-admissible and sub-symbolic heuristics.* In Proceedings of the 16th European Conference on Artificial Intelligence(2004), pages 613-617, 2004.
- Aaron F. Archer. *A modern treatment of the 15-puzzle*. American Mathematical Monthly, 106: pages 793–799, 1999.
- Shahab Jabbari, Sandra Zilles, and Robert Holte. *Bootstrap learning of heuristic functions*. In Proceedings of 3rd Annual Symposium on Combinatorial Search (2010), pages 52–60, 2010.