Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Steps towards a Science of Heuristic Search

Christopher Wilt
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
wilt at *cs.unh.edu*

Abstract

There are many algorithms designed to solve the shortest path problem. Each of the published algorithms has a demonstrated use; a situation in which it is the clear choice. Unfortunately, if faced with a novel problem, there is no reliable robust way to figure out which algorithm should be used to solve the new problem.

When classifying things, the first step is to identify relevant features for classifications. In the context of heuristic search, it not clear what pieces of information should be used to predict search algorithm performance, and the question of algorithm selection for a novel domain is an open question.

We first analyze which domain attributes common algorithms leverage, and discuss how to identify domains containing these attributes. In addition to discussing how to classify domains, we also discuss why the classifications matter for various algorithms. Ultimately, this will allow us to offer more accurate runtime predictions for various algorithms we analyze, allowing us to determine which algorithm will likely offer the best performance.

Introduction

One important area of active research in computer science is heuristic search. Heuristic search problems arise in a variety of contexts, from robot motion planning to video game path finding to logistics. One good example of a heuristic search problem is planning. A simple example of a planning problem is figuring out what route to take to get somewhere, as in GPS route planning. At each intersection, the GPS must decide which road to take, and each road has a different cost associated with it, its length. The objective of a planning problem is to find a sequence of actions that go from the start state to the goal state, while simultaneously minimizing cost. Planning problems are very general, and can be used to represent many different kinds of problems, like routing packages or figuring out how a Mars rover will accomplish its tasks.

In the area of heuristic search, the objective is to find a generic way to solve search problems, like the planning problem discussed above. There are many different heuristic search algorithms including A*, Weighted A*, greedy search, beam search, simplified memory limited A*, iterative deepening A*, and bidirectional search. All of those algorithms solve heuristic search problems, and that sample contains only the algorithms that are popular enough to warrant inclusion in artificial intelligence textbooks. Selecting the proper algorithm is critical. For some problems, the best algorithm will find a solution in under a second, while the worst algorithm won't find a solution at all. When each algorithm was first introduced, it was generally presented with specific examples of domains in which the algorithm performs better than other heuristic search algorithms. This sample is usually small, consisting of only a few domains and a few algorithms. This means that for the vast majority of heuristic search problems, there is no data at all regarding what algorithm should be used to solve a new problem.

My dissertation is about understanding search algorithms, and what different algorithms rely upon to work well, and what can cause the algorithms to perform poorly. In the Towers of Hanoi problem, usually, the cost of moving a disk is 1, but there is no reason the cost has to be 1. Consider a simple modification of the Towers of Hanoi problem where the cost of moving a disk is not 1 but is rather the mass of the disk. What kind of search algorithm should be deployed to solve this particular problem? A number of papers refer to the Towers of Hanoi problem, but in all of these papers, the cost of moving a disk is the same for all disks.

Currently, this kind of problem is solved using trial and error, and due to the complexity of trying different search algorithms on a new problem, the trial and error process is, in practice, often incomplete. My dissertation aims to provide the answer to this kind of question, guiding the algorithm selection process directly to the algorithms most likely to succeed. The first part of my dissertation deals with optimal search algorithms, discussing how to leverage alternative data structures and bidirectional search. The second part of my dissertation deals with suboptimal search, discussing what popular algorithms like Weighted A* and greedy search rely upon for good performance.

Related Work

The literature on heuristic search does not say what the best algorithm to use for solving the modified Towers of Hanoi problem is, and short of trial and error there is no general way to figure out what the best algorithm is for a partic-

ular problem. Other researchers have addressed the question of algorithm performance, but their analyses are generally limited to a specific class of algorithm or problem. For example, Hoffmann (2005) discusses when a enforced hill climbing will find a solution quickly, but only when solving a STRIPS like planning problem. Zahavi et al. (2008) predicts the running time of iterative deepening A* (IDA*) on domains with few duplicates, but knowing how hard IDA* will have to work does not help unless you also know how long other algorithms will take, and ignoring duplicates is a very restrictive assumption, because in almost all heuristic search problems, there is more than one way to get to at least one state (in the navigation example, there is almost always more than one way to go somewhere). Zhang and Korf (1993) discuss the question of which linear space search algorithm is best, comparing depth first branch and bound and recursive best first search, and show when each is best. This kind of analysis is extremely important, and should be extended beyond selecting a linear space search, because linear space searches are rarely used in practice, because other algorithms are almost always faster. Xu et al. (2008) discusses how to construct a boolean satisfability solver through a portfolio approach. Among other things, SATzilla figures out which solvers are likely to work well on the instance at hand, and uses those algorithms. This kind of work is important, and should be done for heuristic search algorithms as well.

Dissertation Outline

My dissertation will be broken up into two main sections. The first section will be about optimal heuristic search, where I discuss algorithms that only return the cheapest solution that exists. Searching backwards from the goal can dramatically reduce the required search effort, if the problem is amenable to that approach. Another area relating to optimal heuristic search where I have done research is data structures. Most heuristic search algorithms require a priority queue, but there are many different ways to create a priority queue, each working best in a specific situation. I describe how to choose a priority queue, as well as how to switch between the queues dynamically for best performance without any a priori knowledge. I intend to finish this section prior to attending AAAI.

The second section of my dissertation will deal with suboptimal heuristic search algorithms. I have already completed one survey of greedy search algorithms, experimenting with a wide variety of greedy search techniques on a suite of six benchmark domains (Wilt, Thayer, and Ruml 2010). That work showed that one feature that makes a huge difference is how much the different actions cost. Problems where all the actions have exactly the same cost could be solved by a variety of common search algorithms, but changing the problem so that some actions cost substantially more than other actions caused the problems to be much more difficult, so much so that many algorithms were unable to find solutions at all. After identifying the operator cost ratio as a relevant feature, the next step was to determine exactly why this feature made problems more difficult, which led to a theorem explaining one reason why these problems are more difficult to solve, (Wilt and Ruml 2011). The other conclusion of this analysis was that algorithms that use estimates of how many actions need to be used to get to the goal are able to deal with the varying action costs. I will spend part of the next academic year figuring out why this information is so useful. This work will form the first subsection of the part of my dissertation on suboptimal search.

One of the most popular suboptimal search algorithms, Weighted A*, will sometimes fail catastrophically. In my dissertation I will explain why the correlation between two properties of states, named h (the estimated cost of getting to a goal) and d* (the number of states on the shortest path to a goal), could be used to predict if weighted A* would fail (Wilt and Ruml 2012). The cause of the correlation (or lack thereof) is not clear, and in the upcoming year, I will analyze the relationship between h and d*, figuring out both why some domains have a strong relationship between those quantities, and why that relationship is so important to some algorithms, but not to others.

Conclusion

One of the most important aspects of science is understanding when a given technique should be used. Predicting when a particular heuristic search algorithm is an appropriate choice is a difficult problem that is generally solved by trial and error. What is needed is a precise science that will replace this difficult and time consuming process.

With an understanding of which domain features are leveraged by the various algorithms, we will be able to look at new problems and identify whether or not they satisfy the implicit assumptions the various algorithms make, ultimately allowing us to identify promising algorithms without the difficult trial and error process that currently drives algorithm selection.

References

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artifial Intelligence Research* 24:685–758.

Wilt, C., and Ruml, W. 2011. Cost-based heuristic search is sensitive to the ratio of operator costs. In *Proceedings of the Fourth Symposium on Combinatorial Search*.

Wilt, C., and Ruml, W. 2012. When does weighted A* fail? In *Proceedings of the Fifth Symposium on Combinatorial Search*.

Wilt, C.; Thayer, J.; and Ruml, W. 2010. A comparison of greedy search algorithms. In *Proceedings of the Third Symposium on Combinatorial Search*.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Intell. Res. (JAIR)* 32:565–606.

Zahavi, U.; Feiner, A.; Burch, N.; and Holte, R. C. 2008. Predicting the performance of IDA* with conditional distributions. In *AAAI'08: Proceedings of the 23rd National Conference on Artificial Intelligence*, 381–386. AAAI Press.

Zhang, W., and Korf, R. E. 1993. Depth-first vs. best-first search: New results. In *Proceedings of AAAI-93*, 769–775.