4511W Writing 2

Nadya Postolaki

February 2020

1 *A Comparison of Greedy Search Algorithms*- A Paper Study

Written by Christopher Wilt, Jordan Thayer, and Wheeler Ruml, *A Comparison of Greedy Search Algorithms* delves into three types of greedy search algorithms- best-first search, hill-climbing search, and beam search- each with respective sub-categories of searches to be tested under the terms of six benchmark domains: path finding in grids, the traveling salesman problem, dynamic robot path finding, the sliding tile puzzle, the pancake puzzle, and the vacuum robot domain.

Best-first, defined by a node ranking function which considers cost of arriving at a node as well as the estimated cost of reaching a goal from a node, had the following search algorithms considered: Weighted A*, greedy search, Window A*, and Multi-state commitment search (MSC-kwA*). These searches are least oriented towards solving a problem as quickly as possible, so time is not of the biggest concern for these algorithms. They are also complete, or can be made complete, which can expend effort to prove the quality of their solutions, and is a result of having an open list of unbounded size and only terminate when a solution has been found or they have emptied the entire open list. All of which was mentioned in the paper, completeness allows for the searches to avoid problems created by heuristic error and local extreme.[5]

Of the best-first algorithms, it was found that Weighted A* and greedy search were the best in terms of the six benchmark domains domains. Weighted A* was able to find the highest quality solutions, while greedy search was usually able to find a solution faster, albeit in a lower quality than that of weighted A*. This is due to greedy best-first searches trying to expand the node that is closest to the goal which increases the likelihood of being led to the solution quicker, therefore evaluating nodes by using the heuristic function $f(n) = h(n)$, whereas A* search evaluates nodes by combining $g(n)$ and $h(n)$ to get the function $f(n) = g(n) + h(n)$, meaning that the first node to try is that with the lowest $g(n) + h(n)$ value.[3] [2]

Hill-climbing search had the following search algorithms considered: Enforced hill-climbing search, real-time search (LSS-LRTA*), and depth-first branch and bound search. Hill-climbing searches expand the most promising descendant of the most recently expanded

node until a solution is encountered- expanding one node at a time. Of the hill-climbing searches, it was found that LSS-LRTA* outperforms enforced hill-climbing and is far more robust. Enforced hill-climbing searches is based on the commonly used hill-climbing algorithm for local searches, but instead uses breadth-first search from the global optimum to find a sequence of actions leading to a heuristically better successor if none are present in the immediate vicinity. [4]

LSS-LRTA* uses A* to determine its local search spaces and searches outward for a fixed duration, committing to the best looking state on the frontier, afterwards running Dijkstra's algorithm to improve heuristic values.[1] Since LSS-LRTA* doesn't go through many nodes at once as the breadth-first enforced hill-climbing algorithm would, it saves the cost of being run due to considering only the best looking states. An issue with hill-climbing algorithms, however, lies in the possibility of continuously running in what seems to be the best case but ending up in a loop or terminating due to getting stuck in a situation in which it can no longer solve the problem- a result of early commitment.

Beam searches exist in between hill-climbing and best-first- exploring a fraction of the space and pruning regions that appear 'worse'. These searches are bounded in memory consumption and therefore have an advantage with better scaling behavior. The two categories of beam searches are best-first and breadth-first. The difference between the two are that best-first beam search is run just like a best-first search, except when the open lists grow beyond the predetermined size limit, the worse of the nodes are removed from the open list until the list is within its size limit, and breadth-first search expands a fixed number of nodes while the rest are removed.[3] Breadth-first was found to be the most effective beam search due to the nature of error in admissible heuristics.

The paper was able to describe the tests run and results clearly in the form of graphs, tables, and figures, which I found most convenient, however may be difficult to understand for someone with little to no prior knowledge of the tests or even the algorithms being tested as little information was given on those aspects. Should the paper be written for a novice in such a study, it would be best to go further in depth about what the searches entailed, as well as *why* these particular algorithms were chosen to test over others.

References

[1] *On the Optimization of Dijkstra's Algorithm*, Lecture Notes in Electrical Engineering, American University of the Middle East, Egaila, Kuwait, 2011. Springer-Verlag Berlin Heidelberg.

[2] Stuart J. Russell, Peter Norvig. Artificial intelligence. In Marcia Horton, editor, *A Modern Approach Third Edition*, chapter 3, pages 64–119. Pearson Education, New Jersey, 2010.

[3] Stuart J. Russell, Peter Norvig. *Artificial Intelligence: A Modern Approach Third Edition.* Pearson Education, 2010.

[4] Andrew Coles and Amanda Smith. Enforced hill climbing search. Technical report, Carnegie Mellon University, School of Computer Science, 1 2007. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume28/coles07a-html/node5.html.

[5] Henry Farreny. Completeness and admissibility for general heuristic search algorithms-a theoretical study: Basic concepts and proofs. *Journal of Heuristics*, 5:353, 1999.