Andrew J Miller
Udacity – Artificial Intelligence Nanodegree
July 12th, 2017

Heuristic Analysis

1. Summary:

For my project I chose to evaluate the following search algorithms: breadth first search, depth first graph search, greedy best first graph search, ignore preconditions heuristic, and the levelsum heuristic. By far, the overall best performing algorithm by speed alone was greedy best first search. However, the ignore preconditions heuristic performed the best if you consider only the strategies that picked the best planning length. The levelsum heuristic and the greedy best first search were on par with one and other on the nodular parameters. However, in my opinion, the levelsum heuristic performed the best consistently on the nodular level.

2. Data/ Results:

Breadth First Search				
Problem 1				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
43	56	180	6	0.0275s
Problem 2				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
3315	4572	29722	9	11.332s
Problem 3				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
14501	17956	127260	12	85.368s

Fig A. Breadth First Search

Depth First Graph Search				
Problem 1				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
12	13	48	12	0.009s
Problem 2				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
1417	1418	11792	452	4.132s
Problem 3				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
4927	4928	39990	1450	27.474s

Fig B. Depth First Graph Search

2. Data/ Results (continued):

Greedy Best First Graph Search				
Problem 1				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
7	9	28	6	0.006s
Problem 2				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
502	504	4415	23	1.101s
Problem 3				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
338	340	2908	18	0.864s

Fig C. Greedy Best First Graph Search

Ignore Preconditions				
Problem 1				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
41	43	170	6	0.037s
Problem 2				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
1428	1430	12934	9	3.611s
Problem 3				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
4693	4695	41464	12	15.154s

Fig D. Ignore Preconditions Heuristic

Levelsum				
Problem 1				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
11	13	50	6	1.677s
Problem 2				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
155	157	1419	9	483.701s
Problem 3				
Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
366	368	3324	12	1713.714s

Fig E. Levelsum Heuristic

3. Discussion:

At an initial glance the results can be somewhat puzzling but after further thought it can be determined that the lower performances of our heuristic searches can be attributed to the fact that they are more computationally expensive. The especially odd results may also be due to the optimization type of hardware. The hardware has been optimized to run the graphics processor in parallel with the central processor. This is possibly handing out a better score to simple computations and damaging the speed of longer arithmetic units. That is, to my understanding, the point of the software version that my machine is optimized for.

Because of the processing's favoritism toward simplistic processing: greedy best first search and depth first search are favored on the testing equipment. While they may not find the best solution, their algorithms are simple point calculations that require very little memory. In any case, greedy best first search did not perform much worse in comparison to finding the best solution. On average, on the testing machine, greedy best first search will perform by far the fastest with a trade-off of a longer planning length. It is however possible that when longer planning lengths become available, that the other methods will become more favored. If greedy best first search gets stuck along a set of bad paths it is very possible that it would perform exceptionally worse. However, in my test cases, this never occurred.

However, when not considering speed, the levelsum heuristic will pick the most efficient pathway faster than any of the other methods. The only issue is the length of the computational unit and the specific example of greedy best first search that is shown. On average, levelsum will pick a better path than greedy best first search, that's its entire purpose, so this isn't very insightful. If levelsum were implemented in a way that decreased its computational expensiveness, or if it was run on a machine optimized for its type of calculation, it would have performed much better. But it all depends on what we're trying to discover and find out. Each method has its advantages and disadvantages. What matters most is the scenarios that they are used within.