Nick Garvey
Nov 11 2017
AI Nanodegree - Heuristic Analysis

Data:

problem	method	expansions	plan_length	time
1	breadth_first_search	43	6	0.16
1	depth_first_graph_search	21	20	0.08
1	uniform_cost_search	55	6	0.21
1	astar_search with h_ignore_preconditions	41	6	0.16
1	astar_search with h_pg_levelsum	11	6	0.51
2	breadth_first_search	3343	9	58.7
2	depth_first_graph_search	624	619	11.36
2	uniform_cost_search	4603	9	75.19
2	astar_search with h_ignore_preconditions	1310	9	21.19
2	astar_search with h_pg_levelsum	74	9	37.74
3	breadth_first_search	14663	12	341.69
3	depth_first_graph_search	408	392	8.17
3	uniform_cost_search	16961	12	336.45
3	astar_search with h_ignore_preconditions	4422	12	88.02
3	astar_search with h_pg_levelsum	229	13	156.31

Optimal Paths:

Problem 1: Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)

Problem 2: Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)

Problem 3: Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)

Analysis:

A few things stand out in the above data.

First, astar_search with h_pg_levelsum took the longest on problem 1 even though it did well on the others. This is likely due to the overhead of constructing the planning graph object. Problem 1 is simple enough that the overhead for this object construction may outweigh the time to solve it.

Second, depth_first_graph_search found a solution for problem 3 extremely quickly. I'd want to see it duplicate this result a few times before making any conclusions here, it might have just gotten lucky.

uniform_cost_search stands out as a poor choice. It was slow for all problems, and required a lot of node expansions. Section 3.4.2 of Artificial Intelligence, a Modern Approach 3rd Edition by Stuart J. Russell and Peter Norvig mentions that cost checks are applied at every node expansion. As all paths are of equal cost, this means that all of these cost checks are useless, and this ends up being an expensive breadth first search.

It is interesting to see all astar methods find optimal methods except for astar_search with h_pg_levelsum. This doesn't suggest a bug in the implementation, Section 10.3.1 of the textbook clearly states that this heuristic is not admissible, and therefore may find an unoptimal solution. However the textbook is also correct that the heuristic is good - it found a solution 1 off of optimal with by far the fewest node expansions.

The best algorithm in practice will depend on the cost trade-off between a suboptimal plan and the computational resources used to find the plan. If a suboptimal plan is fine (e.g. real-time route selection in a self driving car), then astar_search with h_pg_levelsum seems to be the best option.

If the plan is for something more expensive, like air cargo, then taking the time to find an optimal solution is obviously worth it, and something like astar_search with h_ignore_preconditions would be a better choice.