HEURISTIC ANALYSIS

For a Planning Search Agent

Submitted by
Matt Toledo
As part of the AIND Program, Udacity

TABLE OF CONTENTS

Results

air_cargo_p1						
Algorithm	**Node Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time Elapsed (Seconds)**	**Optimality**
Breadth First Search	43	56	180	6	0.058	Yes
Breadth First Tree Search	1458	1459	5960	6	1.498	Yes
Depth First Graph Search	21	22	84	20	0.021	No
Depth First Limited Search	101	271	414	50	0.128	No
Uniform Cost Search	55	57	224	6	0.074	Yes
A* h1	55	57	224	6	0.073	Yes
A* h_ignore_preconditions	41	43	170	6	0.054	Yes
A* h_pg_level_sum	11	13	50	6	0.647	Yes

Optimal Plans		
Uniform Cost Search	**A* h_ignore_preconditions**	**A* h_pg_level_sum**
Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)

air_cargo_p2						
Algorithm	**Node Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time Elapsed (Seconds)**	**Optimality**
Breadth First Search	3343	4609	30 509	9	18.886	Yes
Breadth First Tree Search					TIMED OUT	
Depth First Graph Search	624	625	5602	619	5.436	No
Depth First Limited Search	222 719	2 053 741	2 054 119	50	1628.895	No
Uniform Cost Search	4853	4855	44 041	9	26.193	Yes
A* h1	4853	4855	44 041	9	26.220	Yes
A* h_ignore_preconditions	1450	1452	13 303	9	8.729	Yes
A* h_pg_level_sum	86	88	841	9	51.481	Yes

Optimal Plans		
Uniform Cost Search	**A* h_ignore_preconditions**	**A* h_pg_level_sum**
Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

air_cargo_p3						
Algorithm	**Node Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time Elapsed (Seconds)**	**Optimality**
Breadth First Search	14 120	17 673	124 926	12	97.408	Yes
Breadth First Tree Search					TIMED OUT	
Depth First Graph Search	292	293	2388	288	2.296	No
Depth First Limited Search					TIMED OUT	
Uniform Cost Search	18 223	18 225	159 618	12	126.911	Yes
A* h1	18 223	12 225	159 618	12	1277.171	Yes
A* h_ignore_preconditions	5040	5042	44 944	12	38.125	Yes
A* h_pg_level_sum	325	327	3002	12	261.756	Yes

Optimal Plan		
Uniform Cost Search	**A* h_ignore_preconditions**	**A* h_pg_level_sum**
Load(C1, P1, SFO)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
Load(C2, P2, JFK)	Fly(P2, JFK, ORD)	Fly(P2, JFK, ORD)
Fly(P1, SFO, ATL)	Load(C4, P2, ORD)	Load(C4, P2, ORD)
Load(C3, P1, ATL)	Fly(P2, ORD, SFO)	Fly(P2, ORD, SFO)
Fly(P2, JFK, ORD)	Unload(C4, P2, SFO)	Load(C1, P1, SFO)
Load(C4, P2, ORD)	Load(C1, P1, SFO)	Fly(P1, SFO, ATL)
Fly(P2, ORD, SFO)	Fly(P1, SFO, ATL)	Load(C3, P1, ATL)
Fly(P1, ATL, JFK)	Load(C3, P1, ATL)	Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)	Fly(P1, ATL, JFK)	Unload(C4, P2, SFO)
Unload(C3, P1, JFK)	Unload(C3, P1, JFK)	Unload(C3, P1, JFK)
Unload(C2, P2, SFO)	Unload(C2, P2, SFO)	Unload(C2, P2, SFO)
Unload(C1, P1, JFK)	Unload(C1, P1, JFK)	Unload(C1, P1, JFK)

Analysis

For the three problems above there were two different types of algorithms used. The first of which are those that are non-heuristic search algorithms. These include: Breadth First Search(BFS), Breadth First Tree Search(BFTS), Depth First Graph Search(DFGS), Depth Limited Search(DLS), Uniform Cost Search(UCS) and A* with h1 search(A*h1). Among these algorithms only DFGS and DLS failed to find the optimal solution.

The reason for this being that these two algorithms search down a branch until the end of that branch before considering any other branches[1]. This means that even if it finds a goal state, it's not guaranteed to be the optimal path as a shorter path to the goal could be elsewhere in the tree. Worse yet, if the problem has an infinite state space, Depth First algorithms are not guaranteed to find a solution[2].

The algorithms that returned an optimal path on the other hand behave differently. BFS and BFTS for example prioritize expanding the shortest path first[3]. This guarantees the optimal path is found[1]. As long as a solution exists at some finite level, both of these algorithms are guaranteed to find the solution even in an infinite state space[2].

As for the last two non-heuristic search algorithms, because of the way the problems were set, they both behave similarly to the Breadth First algorithms. Normally UCS expands the path with the lowest combined cost first[4]. However since heuristics weren't being used for this set of algorithms, UCS treated all the nodes as having the same cost. Ties were then decided by taking the shortest path. The same is true with A*h1 as the value for each node is set to one.

Out of these six algorithms, the best performing non-heuristic algorithm was BFS. It expanded the least number of nodes and ran the fastest out of all the non-heuristic algorithms that found a solution.

In regards to the two heuristic functions, A* h_ignore_precondition(A*HIP) and A* with pg_level_sum(PGLS), they both found the optimal solution. However they differ in the way they arrived to their optimal path. A*HIP returned an optimal path faster than A*PGLS for all three problems. This is because by ignoring the preconditions we are able to identify the absolute minimum number of actions required to reach the goal. The fact that this is a fairly simple heuristic makes it faster to compute but makes us expand more nodes[5].

A*PGLS on the other hand is a more complex heuristic as it builds a planning graph and uses that to then solve the problem. It gives us a better estimated cost by decomposing the problem into smaller subgoals and assuming that each subgoal can be solved independently. The heuristic then returns the sum of the level costs from the state we are in to the first level in which all the subgoals are achieved[6]. The result is a heuristic that expands fewer nodes but takes longer to compute.

In conclusion, the best performing algorithm was A* with h_ignore_preconditions. Not only was it fastest out of all the algorithms that found the optimal solution, it also expanded few nodes out of all the non-heuristic algorithms.

References

[1] Artificial Intelligence Nanodegree Program, Udacity - Lesson 10 Search: Search Comparison 1

[2] Artificial Intelligence Nanodegree Program, Udacity - Lesson 10 Search: Search Comparison 3

[3] Artificial Intelligence Nanodegree Program, Udacity - Lesson 10 Search: Breadth First Search 1

[4] Artificial Intelligence Nanodegree Program, Udacity - Lesson 10 Search: Uniform Cost Search

[5] Artificial Intelligence Nanodegree Program, Udacity - Lesson 10 Search: Sliding Blocks Puzzle 2

[6] Artificial Intelligence: A Modern Approach, 3rd Edition, Peter Norvig & Stuart J Russell - Chapter 10.3 Planning Graphs