Udacity Artificial Intelligence Nanodegree

Planning Heuristic Analysis

Research Review

Non-heuristic Search Results

Search Algorithm	Problem	Plan Length	Nodes Expanded	Goal Tests	Time	Optimal
Breadth First Search	1	6	43	56	0.0558	True
Breadth First Search	2	9	3343	4609	18.623	True
Breadth First Search	3	12	14663	18098	114.387	True
Depth First Graph Search	1	20	21	22	0.0294	True
Depth First Graph Search	2	619	624	625	5.947	False
Depth First Graph Search	3	392	408	409	4.0291	False
Uniform Cost Search	1	6	55	57	0.0727	True
Uniform Cost Search	2	9	4852	4854	29.330	True
Uniform Cost Search	3	12	18235	18237	142.17	True

The shortest plan length seems to be coming out of Breadth First Search and Uniform Cost Search where breadth first search seems to be slightly faster in execution time in comparison to Uniform Cost Search. However, UCS seems to expand more nodes and goal tests. Depth First Graph Search is significantly faster that the two other algorithms, however with 2 un-optimal results out of 3, does not seem to be reliable option for this set of problems.

A* With Heuristics Search Results

A* Heuristic	Problem	Plan Length	Nodes Expanded	Goal Tests	Time	Optimal
h_ignore_preconditions	1	6	41	43	0.0777	True
h_ignore_preconditions	2	9	1450	1452	9.283	True
h_ignore_preconditions	3	12	5040	5042	43.142	True
h_pg_levelsum	1	6	11	13	1.262	True
h_pg_levelsum	2	9	86	88	274.75	True
h_pg_levelsum	3	12	318	320	1199.93	True

By comparing the A* search algorithm with two heuristics, “levelsum” is significantly slower due the high cost of calculating the heuristic but with similar results to “ignore preconditions”. In the case of Problem 3, “levelsum” is ~27 times slower. They both come up with plans with the same length but as demonstrated in the table above, the nodes expanded by the “ignore preconditions” heuristic is significantly higher than “levelsum”.

Out of all the algorithms mentioned above, A* with “Ignore Preconditions” heuristic seems to be the most efficient as it expands a large number of nodes in addition to quick execution time. The simplicity of the heuristic implementation due to reduction in preconditions is responsible for its faster execution and it helps with reducing the expansion of ignored nodes overall (Lesson 14.19).

Optimal Solution For Problem #1

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Optimal Solution For Problem #2

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, 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)

Optimal Solution For Problem #3

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)