Denis Silva
Udacity - AI Nanodegree

Implementing a planning search

Problems Optimized Solutions and Comments

Problem 1

After working on the problem 1 and tested with all available algorithms, have checked that the best result was with Greedy Best First Graph Search, with the optimized route:

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

The following table we have the metrics for all algorithm that has been used to verify what was the best option for this problem.

Table 1 - Comparison between Algorithms for Problem 1

Algorithm	Expansions	Goal Tests	New Nodes	Time Elapsed (s)
BFS	43	56	180	0.0287
BFTS	1458	1459	5960	0.8502
DFGS	12	13	48	0.0069
DLS	101	271	414	0.0801
Uniform Cost Search	55	57	224	0.0350
Recursive BFS	4229	4230	17029	2.8471
Greedy BFS	7	9	28	0.0043
A* Search	55	57	224	0.0324

Algorithm	Expansions	Goal Tests	New Nodes	Time Elapsed (s)
A* Search (ignore pre-conditions)	41	43	170	0.0333
A* Search (pg levelsum)	55	57	224	1.6271

Problem 2

For the second problem where more variables have been added it's easy to verify the amount of time that the application had to calculate the optimized plan that is composed by 9 steps that resulted from the A* Search, ignoring pre-conditions, tried to run the A* search with pg levelsum but after 30 minutes the task didn't finish:

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)

The following table we have the metrics for all algorithm that has been used to verify what was the best option for this problem.

Table 2 - Comparison between Algorithms for Problem 2

Algorithm	Expansions	Goal Tests	New Nodes	Time Elapsed (s)
BFS	3343	4609	13050980	9.6000
DFGS	582	583	5211	2.8392
Greedy BFS	998	1000	8982	2.6684
A* Search	4853	4855	44041	11.3173
A* Search (ignore pre-conditions)	1450	1452	13303	4.2523
A* Search (pg levelsum)	*	*	*	*

Problem 3

For the third problem where the complexity is higher than others problems, only some results where possible to be evaluated due computational capacity available and as said at AIMA, "*Many planning problems have* 10^{100} *states or more, and relaxing the actions does nothing to reduce the number of states…* ". So an idea to execute this problem quicker is to relax it using state abstraction. So it was ran the process with the algorithms BFS, A* Search and A* Search (ignore pre-conditions). And between the evaluated algorithms the one that presented to be the most effective was the A* Search ignoring pre-conditions having 3 times less expansions than BFS and being executed in half of the time. The optimized route is:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Fly(P1, ATL, JFK)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

The following table we have the metrics for all algorithm that has been used to verify what was the best option for this problem.

Table 3 - Comparison between Algorithms for Problem 3

Algorithm	Expansions	Goal Tests	New Nodes	Time Elapsed (s)
BFS	14663	18098	129631	39.9248
A* Search	18223	18225	159618	53.3884
A* Search (ignore pre-conditions)	5040	5042	44944	16.5024