Udacity AI Nanodegree

Analysis of Algorithms for Planning Module Project

Omkar Vedpathak, April 2017 cohort

Optimal Plans

Problem 1

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

6 actions

Problem 2

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

9 actions

Problem 3

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

12 actions

Non-heuristic search metric comparison

Out of the non-heuristic search methods only Breadth First Search (BFS), BFS Tree and Uniform Cost search are Optimal, the rest being non-optimal. This is because each of these algorithms will traverse all the required nodes in the search space until the best goal is found.

The Depth First Graph search trades optimality for time for execution, since it only expands one possible path until completion (going as deep as it can), making it the fastest algorithm in almost every Problem. It also expands by far the fewest nodes of all the algorithms, and generates the fewest nodes, making it ideal if availability of storage is limited. On the other end of the spectrum, the BFS Tree algorithm searches the greatest number of nodes out of the non-heuristic algorithms, and fails to complete execution under 10 minutes for Problem 3.

The best performing algorithm overall out of the non-heuristic search algorithms is the uniform cost search, since it combines optimality with a quicker speed of execution vs the other optimal algorithms, however it requires an expansion of a greater number of nodes vs BFS, so would require greater memory requirements.

Heuristic search metric comparison

The A* h_1 and hg_pg_levelsum searches are optimal, however the h_ignore_preconditons search is not always optimal. This is because A* will only find the lowest cost path if the heuristic always gives an estimate that is lower than the true cost of the path, i.e. it is admissible. In the case of ignoring preconditions, the heuristic is not admissible. Removing the constraints and preconditions causes the A* algorithm to traverse paths that are irrelevant to achieving the goal at hand. The h_ignore_preconditions heuristic however does converge to the solution far faster than the other heuristics and expands a smaller number of nodes.

The h_1 heuristic A* search is faster than the hg_pg_levelsum, since using hg_pg_levelsum there is additional time complexity of searching d * n (where d is depth of the planning graph, and n is the number of nodes per level of the planning graph) over the standard A* search, which causes the more complex problems 2 & 3 to run significantly slower with the hg_pg_levelsum heuristic. hg_pg_levelsum does however have one benefit over the h_1, in that it requires a far smaller number of node expansions. In my implementation adding some sort of caching of the hg_pg_levelsum heuristic result for a given planning graph state, would speed up the peformance of using the heuristic, however I have not implanted this caching, as I idn't want to modify the signature of the helper methods calling the heuristic functions.

The best heuristic used in these problems is the h_1 heuristic, i.e. no real heuristic. This is because there is no real advantage with using hg_pg_levelsum over h_1 for this problem set.

Uniform Cost search performs very similarly to the h_1 search, since the A* search algorithm use Uniform Cost Search plus a heuristic to come to its overall cost function, and since the h_1 heuristic is a constant, it performs almost identically to Uniform Cost Search. However since there is the additional memory and processing overhead with having to call the heuristic function, the A* with h_1 actually runs slightly slower than the Uniform Cost Search.

Depending on what we want to optimize for, i.e. memory usage vs speed of execution vs optimality, the best algorithms + heuristic for this problem set are (respectively):

Depth First Graph Search (memory), Depth First Graph Search (memory + speed), Uniform Cost Search (optimality). The Greedy Best First search, although not optimal, is a good compromise between Depth First Graph Search and Uniform Cost Search, having an almost optimal solution, and a relatively fast speed of execution and lower # node expansions required.

Performance of algorithms used in this problem set

Air cargo problem 1

Algorithm	**Node Expansions required**	**# goal tests**	**New Nodes**	**Time elapsed (s)**	**Optimality of solution (optimal plan length)**
BFS	43	56	180	0.036456	6
BFS Tree	1458	1459	5960	0.902404	6
Depth First Graph Search	12	13	48	0.013885	12
Depth Limited Search	101	271	414	0.114165	50
Uniform Cost Search	55	57	224	0.040276	6
Recursive Best First Search	4229	4230	17029	2.896341	6
Greedy Best First Search	7	9	28	0.006571	7
A* Search h_1	55	57	224	0.042223	6
A* Search h_ignore_preconditions	7	9	28	0.005272	6
A* Search hg_pg_levelsum	11	13	50	1.5994232	6

Air cargo problem 2

Algorithm	Node Expansions required	# goal tests	New Nodes	Time elapsed (s)	Optimality of solution (optimal plan length)
BFS	3343	1110	4609	14.423556	9
BFS Tree				> 10 minutes	
Depth First Graph Search	1669	1670	14863	14.0303	1444
Depth Limited Search	222719	2053741	2054119	958.884	50
Uniform Cost Search	4853	4855	44041	11.7437266	9
Recursive Best First Search				>>10 minutes	
Greedy Best First Search	998	1000	8982	2.587667	21
A* Search h_1	4853	4855	44041	13.841	9
A* Search h_ignore_preconditions	998	1000	8982	3.31996259	21
A* Search hg_pg_levelsum	86	88	841	206.913575	9

Air cargo problem 3

Algorithm	Node Expansions required	# goal tests	New Nodes	Time elapsed (s)	Optimality of solution (optimal plan length)
BFS	14663	18098	129631	102.89372	12
BFS Tree				> 10 minutes	
Depth First Graph Search	592	593	4927	3.0456982	571
Depth Limited Search				> 5 minutes	
Uniform Cost Search	18221	18223	159599	51.635550	12
Recursive Best First Search				>10 minutes	
Greedy Best First Search	5560	5562	49015	16.87312	22
A* Search h_1	18221	18223	159599	60.421538	12

A* Search h_ignore_preconditions	5560	5562	49015	16.9347249	22
A* Search hg_pg_levelsum	315	317	2902	4395.119	12