Heuristic Analysis

Optimal Plans

Problem 1: An optimal plan produced by BFS is:
Plan length: 6
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 2: An optimal plan produced by BFS is:
Plan length: 9
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)

Problem 3: An optimal plan produced by BFS is:
Plan length: 12
Load(C2, P2, JFK)
Load(C1, P1, SFO)
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)

Results and Analysis

		Problem 1				
Num	Algo Name	Expansions	Goal Tests	New Nodes	Length	Time (sec)
1	breadth_first_search	43	56	180	6	0.04
2	breadth_first_tree_search	1458	1459	5960	6	1.18
3	depth_first_graph_search	12	13	48	12	0.01
4	depth_limited_search	101	271	414	50	0.12
5	uniform_cost_search	55	57	224	6	0.05
6	recursive_best_first_search with h_1	4229	4230	17029	6	3.49
7	greedy_best_first_graph_search with h_1	7	9	28	6	0.01
8	astar_search with h_1	55	57	224	6	0.06
9	astar_search with h_ignore_preconditions	41	43	170	6	0.05
10	astar_search with h_pg_levelsum	11	13	50	6	0.69
		Problem 2				
Num	Algo Name	Expansions	Goal Tests	New Nodes	Length	Time (sec)
1	breadth_first_search	3343	4609	30509	9	15.94
2	breadth_first_tree_search	N/A	N/A	N/A	N/A	N/A
3	depth_first_graph_search	582	583	5211	575	3.80
4	depth_limited_search	N/A	N/A	N/A	N/A	N/A
5	uniform_cost_search	4852	4854	44030	9	15.75
6	recursive_best_first_search with h_1	N/A	N/A	N/A	N/A	N/A
7	greedy_best_first_graph_search with h_1	990	992	8910	13	3.15
8	astar_search with h_1	4852	4854	44030	9	14.82
9	astar_search with h_ignore_preconditions	1450	1452	13303	9	5.21
10	astar_search with h_pg_levelsum	86	88	841	9	53.62
		Problem 3				
Num	Algo Name	Expansions	Goal Tests	New Nodes	Length	Time (sec)
1	breadth_first_search	14663	18098	129631	12	116.56
2	breadth_first_tree_search	N/A	N/A	N/A	N/A	N/A
3	depth_first_graph_search	627	628	5176	596	3.83
4	depth_limited_search	N/A	N/A	N/A	N/A	N/A
5	uniform_cost_search	18235	18237	159716	12	63.77
6	recursive_best_first_search with h_1	N/A	N/A	N/A	N/A	N/A
7	greedy_best_first_graph_search with h_1	5614	5616	49429	22	20.75
8	astar_search with h_1	18235	18237	159716	12	64.43
9	astar_search with h_ignore_preconditions	5040	5042	44944	12	20.19
10	astar_search with h_pg_levelsum	325	327	3002	12	262.99

Non-heuristics: We choose to compare heuristics 1 (BFS), 3 (DFTS) and 5 (UCS). DFTS is the fastest but does not find an optimal plan for all problems. BFS and UCS are guaranteed to find an optimal path but are slower. In all cases, UCS expands more nodes than BFS; however, UCS is faster since the implementation of the priority queue is more efficient. Note that we notice that in problem 1 UCS is slightly slower than BFS, but we are interested in comparing the efficiency for more complex problems. Therefore problems 2 and 3 are more appropriate benchmark problems to compare algorithm efficiency. In theory, UCS time/space complexity should be larger than BFS; however, due to implementation details, UCS has a smaller time complexity. In particular, BFS stops as soon as it finds a goal, while UCS examines all the nodes at the goal's depth to see if one has a lower cost. Therefore,

UCS does more work by expanding nodes at depth d unnecessarily. Nonetheless, I will recommend the UCS for non-heuristic planning domains since it takes less time using the implementation I was given for this project.

Heuristics: We now compare 9 (IP) and 10 (LS). It's clear that IP and LS heuristics provide significant reduction in the number of nodes expanded. In particular, the LS heuristic provides a more accurate heuristic than the IP heuristic (inferred by the difference in the number of nodes expanded); however, the time to compute the LS heuristic is much larger than the IP heuristic. For problem 3, the IP heuristic is able to expand approximately 2226 nodes per second, while the LS algorithm can only expand approximately 11 nodes per second. This huge reduction in node expansion efficiency outweighs the performance gain in reducing the number of nodes expanded. For these reasons, I recommend the IP algorithm.

Best heuristic: Overall, the ignore preconditions (IP) heuristic was the best for the reasons above. It significantly outperformed non-heuristic search planning methods for the more complex problems, namely problems 2 and 3. For problem 1, it still outperformed all non-heuristic search planning methods in terms of number of nodes expanded; however, it was 0.01 seconds slower than BFS. Overall the IP heuristic has a more substantial cost improvement for more complex problems. For these reasons, I recommend the IP search algorithm.