AIND-Planning - Heuristic Analysis

The code is run on an `AWS c5.xlarge` instance.

Problem 1

Search performance summary:

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	43	56	180	6	0.0218
breadth_first_tree_search	1458	1459	5960	6	0.6715
depth_first_graph_search	21	22	84	20	0.0099
depth_limited_search	101	271	414	50	0.0614
uniform_cost_search	55	57	224	6	0.0259
recursive_best_first_search h_1	4229	4230	17023	6	1.9284
greedy_best_first_graph_search h_1	7	9	28	6	0.0035
astar_search h_1	55	57	224	6	0.0258
astar_search h_ignore_preconditions	41	43	170	6	0.0259
astar_search h_pg_levelsum	11	13	50	6	0.5531
Total	**6021**	**6217**	**24357**	**118**	**3.3272**

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

Problem 2

Search performance summary: (null indicates search takes too long to arrive the goal)

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	3343	4609	30509	9	5.9086
breadth_first_tree_search					
depth_first_graph_search	624	625	5602	619	2.4856
depth_limited_search					
uniform_cost_search	4853	4855	44041	9	8.4991
recursive_best_first_search h_1					
greedy_best_first_graph_search h_1	998	1000	8982	21	1.7043
astar_search h_1	4853	4855	44041	9	8.3634
astar_search h_ignore_preconditions	1450	1452	13303	9	2.9873
astar_search h_pg_levelsum	86	88	841	9	40.2789
Total	**16207**	**17484**	**147319**	**685**	**70.2272**

Optimal plan:
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)

Problem 3

Search performance summary: (null indicates search takes too long to arrive the goal)

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
breadth_first_search	14663	18098	129631	12	29.7623
breadth_first_tree_search					
depth_first_graph_search	408	409	3364	392	1.2326
depth_limited_search					
uniform_cost_search	18236	18238	159726	12	36.9714
recursive_best_first_search h_1					
greedy_best_first_graph_search h_1	5624	5626	49504	22	11.3939
astar_search h_1	18236	18238	159726	12	36.8943
astar_search h_ignore_preconditions	5040	5042	44944	12	11.6611
astar_search h_pg_levelsum	324	326	2993	12	196.0543
Total	**62531**	**65977**	**549888**	**474**	**323.9699**

Optimal plan:
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)

Compare and contrast non-heuristic search result metrics (breadth-first, depth-first, uniform cost)

For breadth-first search and uniform cost search, both of them are able to reach the optimal path for all the three problems (6, 9 and

12). For depth-first search, it is unable to reach the optimal path for all the three problems (20, 619, 392).

In terms of time elapsed, depth-first search outperforms the other two searches and finishes the search the fastest. Breadth-first search needs at least 2x the time to complete the search for Problem 1 and 2, while uniform cost search takes even longer. An even more extreme situation is observed in Problem 3, where depth-first search just takes 1.2 seconds to reach a solution while the other two searches takes more than 30 seconds to find one (though optimal).

In terms of number of node expansions, depth-first search is the most memory efficient, as it just needs to store a tree path at a time. Breadth-first search expands way more nodes and it is even worse in Problem 3 (~38x). Uniform cost search is the worst among the three.

Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum"

A* search guarantees an optimal solution. The `ignore preconditions` heuristic is the fastest among the three heuristics and it is also relatively faster than breadth-first search. Although the `level sum` heuristics expands fewer nodes (~0.06x) than `ignore preconditions`, it takes much longer time (~13-21x) to find the optimal solution as it involves much more computation.

Summary

For non-heuristic search, if the problem requires an optimal solution, breadth-first search is the best search to go for as it guarantees optimal result and performs better (faster and more memory efficient) than uniform cost search. If the problem does not require an optimal solution, or there are other constraints, such as time allowed and memory limit, we should go for depth-first search as it is the fastest to run and requires the least memory.

For heuristic search, the best heuristic used is the one `ignoring preconditions`. As A* search guarantees an optimal solution, we compare it with breadth-first search. For Problem 1, the result is insignificant. For Problem 2 and 3, A* search with `ignore preconditions` takes shorter time (~0.3-0.5x) to reach the goal while also expanding fewer number of nodes (~0.35-0.43x) than breadth-first search. Thus, it is a better option than breath-first search when optimal solution is required. This also shows that often

time a simple heuristic (by relaxing the problem) can reach the goal and performs well.