Heuristic Analysis

Optimal Plan

The following are the optimal plan for the respective air cargo problems.

Air Cargo Problem 1

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

Air Cargo Problem 2

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Air Cargo Problem 3

Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Heuristic vs Non-heuristic Search

The performance of the search algorithms are summarized in the tables below, for every air cargo problem.

Air Cargo Problem 1

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	43	56	180	6	0.02849384967
breadth_first_tree_search	1458	1459	5960	6	0.9056718267
depth_first_graph_search	21	22	84	20	0.01332030682
depth_limited_search	101	270	414	50	0.08436589382
uniform_cost_search	55	57	224	6	0.03449010133
recursive_best_first_search h_1	4229	4230	17023	6	2.647947969
greedy_best_first_graph_search h_1	7	9	28	6	0.009579859421
astar_search h_1	55	57	224	6	0.03682294267
astar_search h_ignore_preconditions	41	43	170	6	0.03721879484
astar_search h_pg_levelsum	11	13	50	6	0.5564997977

Air Cargo Problem 2

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	3343	4609	30509	9	13.25876011
breadth_first_tree_search	#N/A	#N/A	#N/A	#N/A	#N/A
depth_first_graph_search	624	625	5602	619	3.420618999
depth_limited_search	#N/A	#N/A	#N/A	#N/A	#N/A
uniform_cost_search	4853	4855	44041	9	12.11294626
recursive_best_first_search h_1	#N/A	#N/A	#N/A	#N/A	#N/A
greedy_best_first_graph_search h_1	998	1000	8982	21	2.558451009
astar_search h_1	4853	4855	44041	9	12.20717646
astar_search h_ignore_preconditions	1450	1452	13303	9	4.498884345
astar_search h_pg_levelsum	86	88	841	9	51.53916764

Air Cargo Problem 3

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	14663	18098	129631	12	100.3345867
breadth_first_tree_search	#N/A	#N/A	#N/A	#N/A	#N/A
depth_first_graph_search	408	409	3364	392	1.735846112
depth_limited_search	#N/A	#N/A	#N/A	#N/A	#N/A
uniform_cost_search	18223	18225	159618	12	51.56861951
recursive_best_first_search h_1	#N/A	#N/A	#N/A	#N/A	#N/A
greedy_best_first_graph_search h_1	5578	5580	49150	22	15.8892973
astar_search h_1	18223	18225	159618	12	53.92626286
astar_search h_ignore_preconditions	5040	5042	44944	12	17.22840986
astar_search h_pg_levelsum	315	317	2902	12	267.8407635

Discussion

Refer to the 3 tables above for the following analyses.

Exceptional Cases

Breadth first tree search, depth limited search and recursive best first search, took a very long time to run and never found the solution to problem 2 and 3, hence the #N/A in the results. They will not be discussed for problem 2 and 3.

Optimality Guarantee

The path cost is a nondecreasing function of the depth of the node for the air cargo problems, therefore breadth first search and breadth first tree search guarantees optimality. Uniform cost search guarantees optimality because it always expand the node with the lowest path cost.

The heuristic h_1, h_ignore_preconditions are admissible, therefore the A* search algorithms, including the recursive best first search, using these heuristics guarantees optimality. The goals in the air cargo problems are independent, therefore A* search using the h_pg_levelsum guarantees optimality. Note that A* with h_1 heuristic is identical to uniform cost search.

Contrarily, depth first graph search and depth limited search do not guarantee optimality because it might find a suboptimal goal first. Similarly, greedy best first graph search does not allow backtracking, and all visited nodes are considered final, therefore it does not guarantee optimality.

Node Expansions

Recursive best first search expands the most nodes because it has to revisit the nodes repeatedly. Next up is breadth first tree search, which expands the 2nd most number of nodes. In tree search, the visited nodes is not recorded, hence can be revisited multiple times.

For small problems, greedy best first search expands the least nodes because of the small search space. However, this is not the case for larger search space as seen in problem 2 and 3. The lack of backtracking causes greedy BFS to expand more nodes to reach the goals.

A* search using admissible heuristics expands similar number of nodes. Notice that A* h_1 and uniform cost search expands the same number of nodes, because they are operationally identical. Also notice that, any admissible or consistent heuristics will expand no more nodes than h_1 or UCS, because they are basically best-first search algorithm without heuristics.

Depth first search seemingly expands less number of nodes compared to other algorithm because the search terminates as soon as the goals is found, even when the solution is suboptimal. Interestingly, depth limited search expands more nodes than DFS, because the goals along the depth is now being cut off. It has to search more nodes across the specified depth to find the goals.

For large search spaces, the A* planning graph level sum search expand the least number of nodes. This is because planning graph uses different approach than conventional search algorithm. Instead of treating the problem as state space graph, it treats the problem as planning graph with alternating levels of literals and actions, and their respective mutexes. Rather than searching through the graph, the algorithm constructs the planning graph until it has levelled off, then only extracts the solution using boolean constraint satisfaction method.

Time Elapsed

For problem 1, recursive best first search took the longest to find the solution, followed by BF tree search. The 3rd slowest algorithm is A* planning graph search using level sum heuristic. This is because constructing the graph is computationally expensive and time consuming. As for the rest, they performed similarly given the small search space.

Problem 2 and 3 is the better benchmark because of the larger search space. DFS and greedy BFS are the fastest, due to the lack of backtracking (hence giving suboptimal solution). A* with h_1 heuristic and uniform cost search took almost identical time to finish. Similar to the node expansion performance, any admissible or consistent heuristics will perform no slower than h_1 or UCS (except unless the heuristic is extremely complex). Therefore, A* with h_ignore_preconditions is faster than h_1 or UCS.

BFS performs 2nd slowest due to the expansion of nodes depth by depth to find the goals.

The slowest in this case is the A* planning graph search using level sum heuristic. This is because the planning graph is more complex and requires more iterations to level out.

A* with “ignore preconditions” vs “level-sum”

For problems 1, 2, and 3, the level sum planning graph consistently expand significantly less nodes than A* ignore preconditions. As the search space grows larger, the difference in expanded nodes becomes larger. For example, A* ignore preconditions in problem 3 expanded 16 times more nodes than level sum planning graph.

Nevertheless, level sum planning graph performed significantly slower than A* ignore preconditions. It is 11 times faster in problem 2 and 15 times faster in problem 3, which is very substantial in real life applications. One can hypothesis though, if the search space is considerable bigger than these air cargo problems, the level sum planning graph might be faster.

Best Heuristic

The best heuristic would be h_ignore_preconditions, used in A* search because it guarantees optimality and performs fairly fast. Compared to other optimal non-heuristic method, it is significantly faster especially for problems with larger search space.