Eric Iacutone
Heuristic Analysis, Air Cargo Planning

Problem 1

Initial State:

Init(At(C1, SFO) ∧ At(C2, JFK)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))

Optimal Path:

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

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	43	56	180	6	0.033417346
breadth_first_tree_search	1458	1459	5960	6	0.95198957
depth_first_graph_search	12	13	48	12	0.00854612
depth_limited_search	101	271	414	50	0.08851348
uniform_cost_search	55	57	224	6	0.037213593
recursive_best_first_search with h_1	4229	4230	17029	6	2.839088673
greedy_best_first_graph_search with h_1	7	9	28	6	0.005556631
astar_search with h_1	55	57	224	6	0.038135162
astar_search with h_ignore_preconditions	41	43	170	6	0.041177149
astar_search with h_pg_levelsum	11	13	50	6	0.918286929

Problem 2

Initial State:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

Optimal Path:

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)

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	3343	4609	30509	9	15.97970083
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	582	583	5211	575	4.873053137
depth_limited_search	-	-	-	-	-
uniform_cost_search	4852	4854	44030	9	18.89642958
recursive_best_first_search with h_1	-	-	-	-	-
greedy_best_first_graph_search with h_1	990	992	8910	15	4.19024557
astar_search with h_1	4852	4854	44030	9	13.59187847
astar_search with h_ignore_preconditions	1450	1452	13303	9	5.54513472
astar_search with h_pg_levelsum	86	88	841	9	219.0951035

<u>Problem 3</u>

Initial State:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

Optimal Path:

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

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	14663	18098	129631	12	137.5597569
breadth_first_tree_search	-	-	-	-	-
depth_first_graph_search	627	628	5176	596	5.804484381
depth_limited_search	-	-	-	-	-
uniform_cost_search	18235	18237	159716	12	64.4731504
recursive_best_first_search with h_1	-	-	-	-	-
greedy_best_first_graph_search with h_1	5614	5616	49429	22	20.54500021
astar_search with h_1	18235	18237	159716	12	62.11595054
astar_search with h_ignore_preconditions	5040	5042	44944	12	21.31307737
astar_search with h_pg_levelsum	-	-	-	-	-

Results

For simple problems, both breadth-first and A* search both find a solution in a reasonable time period with an equal plan length. As the planning graph complexity increases, the increase in expansions, goal tests, new nodes and time elapsed varies drastically among the search algorithms. The results indicate that a depth-first graph search is the fastest for the problem 3 (contains the most complexity). However, this algorithm also has the longest plan length. Since plan length is import, we can discard depth-first search results. Both breadth-first search and A* search find equal plan lengths for the problem set. However, as complexity grows, A* search finds an optimal path much faster then breadth-first search. Another advantage to A* search is the ability to code an informed heuristic, with the h_ignore_preconditions heuristic reducing the amount of time to find the optimal plan length.

The definition of h_ignore_preconditions is “this heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.” The definition of h_pg_levelsum is “this heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.” I believe the h_pg_levelsum is more a more complex heuristic to calculate; it takes into account all the actions in the planning graph. The heuristic becomes much more memory intensive to perform and therefore slower than h_ignore_preconditions.