Graph Analysis

Optimal Path

Problem 1

The first problem is fairly easy to solve, since all planes are already in the airports where the cargo is, so they just need to load it in the plane, fly to the other airport and unload the cargo. The optimal plan contains six steps, one of the possible optimal plans is shown below:

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

Problem 2

The second problem is a little more difficult because it adds one cargo, one airport and one plane, but it still relatively easy to solve. The optimal plan contains nine steps, an optimal solution is shown below:

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

The third problem is the most complex one, there are only two planes to carry the cargo around, so the route is longer. The optimal plan contains twelve steps, an optimal solution is shown below:

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

Non Heuristic Search Methods

If we look at the no heuristic methods, we can see that the only algorithm that gave us an optimal path is breadth first search. It is quite intuitive that breadth first will give us the shortest path because it will start increasing the path length only if it doesn't find a solution, so it is warranted to find the shortest one; while for depth first search it will increasing the length of the graph until it finds a solution, for example in problem two and three, this algorithm finds a solution that is over 30x longer. In terms of elapsed time though, we find that breadth first search is the most time consuming, since it has to explore all the nodes in one level before going further. Greedy best first search is somewhere in between the other two algorithms, it finds better plans than depth first, but it takes much shorter than breadth first search to find these paths.

Search Method	Problem 1	Problem 2	Problem 3
breadth_first_search	6	9	12
depth_first_graph_search	20	619	392
greedy_best_first_graph_search	6	21	22

Table 1: Plan Length

Search Method	Problem 1	Problem 2	Problem 3
breadth_first_search	43, 56, 180	3.343, 4.609, 30.509	14.663, 18.098, 129.631
depth_first_graph_search	21, 22, 84	624, 625, 5.602	408, 409, 3.364
greedy_best_first_graph_search	7, 9 , 28	990, 992, 8.910	5.614, 5.616, 49.429

Table 2: Nodes (Expansions, Goal Tests, New Nodes)

Search Method	Problem 1	Problem 2	Problem 3
breadth_first_search	0.04s	16.43s	129.95s
depth_first_graph_search	0.02s	4.77s	2.28s
greedy_best_first_graph_search	0.01s	3.00s	21.00s

Table 3: Elapsed Time

Heuristic Search Methods

We can see that all heuristic methods find the an optimal path, but depending on the heuristic used the elapsed time and nodes used varies quite dramatically, we can establish that the best performing heuristic was ignoring the preconditions, it takes us quite quickly to a solution. Pg levels_sum doesn't work at all, in the easiest problem, which with most search methods takes under a second to run, took over 30, despite not many nodes were created, my guess is that it is computationally expensive to create a PlanningGraph object overtime the heuristic has to be computed.

Search Method	Problem 1	Problem 2	Problem 3
astar_search with h_1	6	9	12
astar_search with h_ignore_preconditions	6	9	12
astar_search with h_pg_levelsum	6	NA	NA

Table 4: Plan Length

Search Method	Problem 1	Problem 2	Problem 3
astar_search with h_1	55, 57, 224	4.852, 4.854, 44.030	18.235, 18.237, 159.716
astar_search with h_ignore_preconditions	41, 43, 170	1.450, 1.452, 13.303	5.040, 5.042, 44.944
astar_search with h_pg_levelsum	30, 32, 118	NA	NA

Table 5: Nodes (Expansions, Goal Tests, New Nodes)

Search Method	Problem 1	Problem 2	Problem 3
astar_search with h_1	0.06s	15.35s	67.37
astar_search with h_ignore_preconditions	0.06s	5.97s	26.25s
astar_search with h_pg_levelsum	31.16s	NA	NA

Table 6: Elapsed Time