Air Cargo Planning Problems

In this project we were building a solution for deterministic logistics planning problems for an Air Cargo transport system using planning search agent. Using progression search algorithms optimal plans for each problem were computed. To aid the agent, domain-independent heuristics were implemented.

We were looking at three Air Cargo problems with the same schema, but different initial states and goals. Air Cargo problems were defined in the following schema:

```
Action(Load(c, p, a),
            PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
            EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
            PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
            EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
            PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
            EFFECT: ¬ At(p, from) ∧ At(p, to))
```

Below are initial states and goals of all three problems:

1) Air Cargo Problem 1

```
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))
```

2) Air Cargo Problem 2

```
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))
```

3) Air Cargo Problem 3

```
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))
```

Uniformed Search Strategies Analysis

To solve Air Cargo problems there was a number of uniformed planning searches implemented. We tested all of them in terms of performance (execution time, whether or not they achieved optimal goal and etc.). The optimal paths form problems 1, 2, 3 are of length 6, 9 and 12 respectively. Below we provide and examples of optimal paths for each problem:

1) Problem 1:
   Load(C2, P2, JFK)
   Load(C1, P1, SFO)
   Fly(P2, JFK, SFO)
   Unload(C2, P2, SFO)
   Fly(P1, SFO, JFK)
   Unload(C1, P1, JFK)

2) Problem 2:
   Load(C3, P3, ATL)
   Load(C2, P2, JFK)
   Load(C1, P1, SFO)
   Fly(P3, ATL, SFO)
   Unload(C3, P3, SFO)
   Fly(P2, JFK, SFO)
   Unload(C2, P2, SFO)
   Fly(P1, SFO, JFK)
   Unload(C1, P1, JFK)

3) Problem 3:
   Load(C2, P2, JFK)
   Load(C1, P1, SFO)
   Fly(P2, JFK, ORD)
   Load(C4, P2, ORD)
   Fly(P2, ORD, SFO)
   Unload(C2, P2, SFO)
   Unload(C4, P2, SFO)
   Fly(P1, SFO, ATL)
   Load(C3, P1, ATL)
   Fly(P1, ATL, JFK)
   Unload(C1, P1, JFK)
   Unload(C3, P1, JFK)

Below, there are the results of the tests (note that if test took longer than 10 minutes to complete we cancelled the test).

PROBLEM 1	Expansions	Goal Tests	New Nodes	Time Elapsed (seconds)	Reached Optimal
1. breadth_first_search	43	56	180	0.13	Yes
2. breadth_first_tree_search	1458	1459	5960	3.95	Yes
3. depth_first_graph_search	12	13	48	0.04	No
4. depth_limited_search	101	271	414	0.37	No
5. uniform_cost_search	55	57	224	0.16	Yes

PROBLEM 2	Expansions	Goal Tests	New Nodes	Time Elapsed (seconds)	Reached Optimal
1. breadth_first_search	3401	4672	31049	59	Yes
2. breadth_first_tree_search	-	-	-	-	-
3. depth_first_graph_search	350	351	3142	6.3	No
4. depth_limited_search	-	-	-	-	-
5. uniform_cost_search	4761	4763	43206	50	Yes

PROBLEM 3	Expansions	Goal Tests	New Nodes	Time Elapsed (seconds)	Reached Optimal
1. breadth_first_search	14491	17947	128184	419.8	Yes
2. breadth_first_tree_search	-	-	-	-	-
3. depth_first_graph_search	1948	1949	16253	81.59	No
4. depth_limited_search	-	-	-	-	-
5. uniform_cost_search	17783	17785	155920	238.7	Yes

Out of five search algorithms which were tested, Breadth First Search and Uniform Cost reached optimal solution. If we look at the performance, Depth First Graph Search was the fastest, however it didn't reach the optimal path. If our priority is to find optimal path with the least resources, the potential candidate for the solution would be Uniform Cost Search, because it reached optimal solution in least time for all three problems. On the other hand, if optimal solution is not a priority, then Depth First Graph Search would be an option as it took the least expansions and new nodes as well as memory.

Informed Search Strategies Analysis

In addition to uninformed search strategies, we tested the performance of informed search strategies as well. We looked at how these strategies performed in terms of time taken, memory

usage and other metrics. Below, there are the results (as previously, test which took longer than 10 minutes we cancelled them).

PROBLEM 1	Expansions	Goal Tests	New Nodes	Time Elapsed	Reached Optimal
8. astar_search h_1	55	57	224	0.16	Yes
9. astar_search h_ignore_preconditions	41	43	170	0.15	Yes
10. astar_search h_pg_levelsum	11	13	50	3.8	Yes

PROBLEM 2	Expansions	Goal Tests	New Nodes	Time Elapsed	Reached Optimal
8. astar_search h_1	4761	4763	43206	52	Yes
9. astar_search h_ignore_preconditions	1450	1452	13303	20	Yes
10. astar_search h_pg_levelsum	-	-	-	-	-

PROBLEM 3	Expansions	Goal Tests	New Nodes	Time Elapsed	Reached Optimal
8. astar_search h_1	17783	17785	155920	243.4	Yes
9. astar_search h_ignore_preconditions	5003	5005	44586	73.9	Yes
10. astar_search h_pg_levelsum	-	-	-	-	-

As we can see, all algorithms reached optimal results, except for A* Search with Level Sum heuristic which didn't reach the solution in 10 minutes. Looking at time taken to reach the solution, A* search with Ignore Precondition Heuristic performed the best as long as this method to the least time to reach the optimal solution. However, looking at Problem 1 A* Search with Level Sum heuristic used the least memory as it made the least number of expansions and new nodes so if time taken to reach the solution wasn't a priority.

Comparison of Uninformed and Informed searches

If we compare Uninformed and Informed Search results we can see that A* Search with Ignore Precondition Heuristics outperformed Uniformed Search by time taken as well as memory usage. For this reason A* Search with Ignore Precondition Heuristics would be the best method for Air Cargo Search problems.