Heuristic Analysis – Air Cargo Planning Problem

1 Introduction

In this project, we implement a planning search agent for a deterministic logistics planning problems for an Air Cargo transport system. After defining the air cargo problem and action schema, we first run uninformed planning searches and provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for various search methods (e.g. breadth-first and depth-first search). In the second part of the project, we apply a planning graph to the search problem with automated domain-independent heuristics with A* search. Finally, we compare the results of the domain-independent heuristics against the uninformed planning searches to evaluate the performance of the search methods.

All problems are stated in the following Air Cargo domain:

- Air Cargo Action 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))
```

- Problem 1 initial state and goal (air_cargo_p1):

```
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))
```

- Problem 2 initial state and goal (air_cargo_p2):

```
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))
```

- Problem 3 initial state and goal (air_cargo_p3):

```
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))
```

2 Non-heuristic search methods

In the following, we run uninformed planning searches for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3`; provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm. If depth-first takes longer than 10 minutes, we stop the search and provide this information in the documentation. Test results were generated by using the `run_search` script from the command line:

```
python run_search.py -p 1 -s 1 2 3 4 5 6 7 >> results_p1.txt

python run_search.py -p 2 -s 1 3 5 7 >> results_p2.txt

python run_search.py -p 3 -s 1 3 5 7 >> results_p3.txt
```

In the search script the [-p] defines the problem and [-s] the search method. The search script includes seven non-heuristic search methods to choose from:

1. Breadth_first_search
2. Breadth_first_tree_search
3. Depth_first_graph_search
4. Depth_limited_search
5. Uniform_cost_search
6. Recursive_best_first_search
7. Greedy_best_first_graph_search

We didn’t include breadth first tree search (2), depth limited search (4) and recursive best first search (6) for problem 2 and 3 as the execution time exceeded 10 minutes.

The results for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3` are shown below. Optimal solutions for node expansions required, number of goal tests, time elapsed, and optimality are bold.

For problem 1, we find that breadth first search, breadth first tree search, uniform cost search, recursive best first search, and greedy best first graph search are optimal. Graph search methods such as depth first graph search and greedy best first graph search significantly reduce the number of node expansions and goal test, and therefore search time. As greedy best first search is optimal and minimizes the search speed it counts as the best search method for problem 1.

For problem 2 and 3, only breath first search and uniform cost search are optimal. Breadth first tree search, depth limited search and recursive best first search weren’t considered due to execution time constraints. In both cases, depth first graph search minimizes node expansions, goal tests and search time but isn’t optimal for path length. As breadth first search is optimal and uses less computation time as uniform cost search for the search problem it counts as the best search method for problem 2 and 3. If optimal path length isn’t the primary objective, greedy best first graph search is a good alternative as it reduces search time elapsed.

2.1 Air_cargo_p1 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
Breadth first search	43	56	0.034	**6**	**Yes**
Breadth first tree search	1458	1459	1.059	**6**	**Yes**
Depth first graph search	12	13	0.009	*12*	*No*
Depth limited search	101	271	0.101	*50*	*No*
Uniform cost search	55	57	0.043	**6**	**Yes**
Recursive best first search	4229	4230	3.092	**6**	**Yes**
Greedy best first graph search	**7**	**9**	**0.006**	**6**	**Yes**

2.2 Air_cargo_p2 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
Breadth first search	3343	4609	14.045	**9**	**Yes**
Breadth first tree search	-	-	-	-	-
Depth first graph search	**582**	**583**	**3.126**	*575*	*No*
Depth limited search					
Uniform cost search	4852	4854	45.633	**9**	**Yes**
Recursive best first search	-	-	-	-	-
Greedy best first graph search	990	992	7.366	*15*	*No*

2.3 Air_cargo_p3 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
Breadth first search	14491	17947	98.160	**12**	**Yes**
Breadth first tree search	-	-	-	-	-
Depth first graph search	**1948**	**1949**	**18.604**	1878	No
Depth limited search	-	-	-	-	-
Uniform cost search	17783	17785	362.444	**12**	**Yes**
Recursive best first search	-	-	-	-	-
Greedy best first graph search	4031	4033	67.036	22	No

3 Domain-independent heuristic search methods

In the following, we run A* planning searches using the heuristics we implemented on `air_cargo_p1`, `air_cargo_p2` and `air_cargo_p3`; provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm. If depth-first takes longer than 10 minutes, we stop the search. Test results were generated by using the `run_search` script from the command line:

```
python run_search.py -p 1 -s 8 9 10 >> results_p1_astar.txt

python run_search.py -p 2 -s 8 9 10 >> results_p2_astar.txt

python run_search.py -p 3 -s 8 9  >> results_p3_astar.txt
```

The search script includes three domain-independent heuristic search methods to choose from:

8. Astar_search with h_1
9. Astar_search with h_ignore_preconditions
10. Astar_search with h_levelsum

The results for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3` are shown below.

3.1 Air_cargo_p1 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
A* with h1	55	57	**0.045**	**6**	**Yes**
A* with ignore preconditions	41	43	0.053	**6**	**Yes**
A* with level sum	**37**	**39**	2.558	**6**	**Yes**

3.2 Air_cargo_p2 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
A* with h1	4852	4854	45.732	**9**	**Yes**
A* with ignore preconditions	**1506**	**1508**	**15.144**	**9**	**Yes**
A* with level sum	2347	2349	1535.922	**9**	**Yes**

3.3 Air_cargo_p3 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimality*
A* with h1	18235	18237	388.355	**12**	**Yes**
A* with ignore preconditions	**5118**	**5120**	**95.753**	**12**	**Yes**
A* with level sum	-	-	-	-	-

For domain-independent heuristic A* search we find that for all problems all search method are optimal when concerning the path length. Further we find that for problem 1, astar_search with h_levelsum minimizes node expansion but is slowest when considering search time. For problem 1, astar_search with h1 and astar_search with h_ignore_preconditions are faster than astar_search with h_levelsum where astar_search with h1 is slightly faster. For problem 2 and 3 we can see that astar_search with h_ignore_preconditions is optimal and also minimizes node expansion, goal tests and time elapsed. We didn't include astar_search with h_levelsum for problem 3 as the execution time exceeded 10 minutes.

4 Conclusion

We can conclude that there are both non-heuristic and domain-independent heuristic search methods that provide optimal action plans for our air cargo planning problem. For the non-heuristic search methods we saw that both breadth first search and uniform cost search are optimal. For the domain-independent heuristic search methods all A* search methods are optimal.

When we further consider the execution time, node expansions and goal tests, we saw that for non-heuristic search methods the depth first graph search was fastest but for all problems non-optimal. For domain-independent heuristic search methods, A* search with ignore preconditions heuristic was fastest and optimal with regards to plan length. Therefore, A* search with ignore preconditions heuristics is the preferred strategy for our problem among all search methods.

Our conclusion and results further have shown the advantage of domain-independent heuristic search methods when optimality is a primary concern.