Ryosuke Yamamura
July 30th 2017

Heuristic Analysis – Air Cargo Planning Problem

1 Introduction

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

All problems are stated as following:

<u>Air Cargo Action Schema:</u>

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))

<u>Problem 1 initial state and goal (air_cargo_p1):</u>

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))

<u>Problem 2 initial state and goal (air_cargo_p2):</u>

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, I run uninformed non-heuristic searches for air_cargo_p1, p2, and 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, I stop the search and provide this information in the documentation. Test results Ire 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
python run_search.py -p 2 -s 1 3 5 7
python run_search.py -p 3 -s 1 3 5 7
```

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 following:

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

I didn't include 2, 4, and 6 for problem 2 and 3 due to taking longer 10 minutes to run. Thus, the results for air_cargo_p1, p2, and air_cargo_p3 are shown below.

2.1 Air_cargo_p1 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimal?*
Breadth first search	43	56	0.0343	6	Yes
Breadth first tree search	1458	1459	1.0598	6	Yes
Depth first graph search	12	13	0.0091	12	No
Depth limited search	101	271	0.1099	50	No
Uniform cost search	55	57	0.0422	6	Yes
Recursive best first search	4229	4230	3.1800	6	Yes
Greedy best first graph search	7	9	0.0082	6	Yes

2.2 Air_cargo_p2 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimal?*
Breadth first search	3343	4609	15.6610	9	Yes
Breadth first tree search	–	–	–	–	–
Depth first graph search	1669	1670	15.2937	1444	No
Depth limited search	–	–	–	–	–
Uniform cost search	4852	4854	13.4508	9	Yes
Recursive best first search	–	–	–	–	–
Greedy best first graph search	990	992	2.8603	15	No

2.3 Air_cargo_p3 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimal?*
Breadth first search	14663	18098	115.8941	12	Yes
Breadth first tree search	–	–	–	–	–
Depth first graph search	592	593	3.5030	571	No
Depth limited search	–	–	–	–	–
Uniform cost search	18235	18237	59.0242	12	Yes
Recursive best first search	–	–	–	–	–
Greedy best first graph search	5614	5616	18.5928	22	No

3 Domain-independent heuristic search methods

In the following, I run A* planning searches using the heuristics I 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, I stop the search. Test results Ire generated by using the run_search script from the command line:

```
python run_search.py -p 1 -s 8 9 10
python run_search.py -p 2 -s 8 9 10
python run_search.py -p 3 -s 8 9 10
```

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

8. A* search with h_1
9. A* search with h_ignore_preconditions
10. A* search with h_levelsum

I didn't include A* with h_levelsum for problem 3 due to taking longer 10 minutes to run. Thus, 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*	*Optimal?*
A* with h_1	55	57	0.0431	6	Yes
A* h_ignore_preconditions	41	43	0.0473	6	Yes
A* h_pg_levelsum	23	25	0.8351	6	Yes

3.2 Air_cargo_p2 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimal?*
A* with h_1	4852	4854	13.4188	9	Yes
A* h_ignore_preconditions	1450	1452	4.8572	9	Yes
A* h_pg_levelsum	3337	3339	1764.5640	9	Yes

3.3 Air_cargo_p3 results

Search method	*Expansions*	*Goal Tests*	*Time Elapsed*	*Path Length*	*Optimal?*
A* with h_1	18235	18237	60.6404	12	Yes
A* h_ignore_preconditions	5040	5042	18.9016	12	Yes
A* h_pg_levelsum	10135	10137	>10min	12	–

For domain-independent heuristic A* search, all search method are optimal for path length. Furthermore, A* with h_levelsum minimizes node expansion but it is the slowest search time. For problem 1, A* with h1 and A* with h_ignore_preconditions are faster than A* with h_levelsum where A* with h1 is slightly faster. For problem 2 and 3, A* with h_ignore_preconditions is optimal and also minimizes node expansion, goal tests and time elapsed.

4 Optimal Plan

4.1 Air Cargo Problem 1

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

4.2 Air Cargo Problem 2

```
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)
```

4.3 Air Cargo Problem 3

```
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)
```

5 Conclusion

Both non-heuristic and domain-independent heuristic search methods could provide optimal action plans for air cargo planning problem project. For the non-heuristic search methods, 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 consider the execution time, node expansions and goal tests, the depth first graph search was fastest in non-heuristic search methods but not optimal. A* search with ignore preconditions heuristic was fastest and optimal with regards to plan length in domain-independent heuristic search methods,. Thus, A* search with ignore preconditions heuristics is the best strategy for our problem among all search methods.

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