Udacity AI Nanodegree Course
AI Planning Search – Heuristic Analysis

Satish Boggarapu

This project consisted of defining a group of problems in the classical Planning Domain Definition Language (PDDL) for the air cargo domain. These problems are then set up for search, experimented with various automatically generated heuristics, which include planning graph heuristics to solve each of the three problems. The three problems defined in the project are displayed below.

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))
```

Problem2:

```
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:

```
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))
```

For each of the three problems listed above a non-heuristic planning solution search was done which consisted of a breadth first search, depth first search, and uniform cost search. Along with that another analysis was done which was A* planning search with various heuristics. The metrics for each of the runs will be displayed for each problem along with some analysis below.

Table 1: Air Cargo Problem 1 Heuristics and Non-Heuristics Metrics

	Expansions	Goal Tests	New Nodes	Plan Length	Time(s)
Breadth First Search	43	56	180	6	0.04679
Depth First Graph Search	12	13	48	12	0.01253
Uniform Cost Search	55	57	224	6	0.06436
A* Search with h_ignore_preconditions	41	43	170	6	0.06382
A* Search with h_pg_levelsum	55	57	224	6	1.85041

Air cargo problem 1 non-heuristic searches resulted in some mixed metrics, where certain searches had better metrics for certain categories, but not for all of them. At first glance it looks like depth first graph search had the best results, since it had a significant number of fewer node expansions, goal tests, and new nodes, along with a faster search time. But it doesn't have the shortest path length. On the other hand, breadth first search and uniform cost search had a higher number of node expansions, goal tests, new nodes discovered and a longer search time. But they all had a shortest plan length of 6. Based on this, the best search depends on limited resource available. For example, if memory is a limited resource and time is not a limited resource, the best non-heuristic search would be depth first graph search, since it had fewer number of expansions, goal tests, new nodes discovered and a higher search time. On the other hand, if memory is not a limited resource but time is then breadth first search would be best since it had the fastest search time and shortest path length.

The heuristic results also had similar results. A* search with h_ignore_preconditions, was the best heuristic since it had a significantly faster search time, expanded fewer nodes, performed fewer goal tests, and discovered fewer new nodes then A* search with h_pg_levelsum. Both heuristics had the same plan length, but significantly different search times. Below are the solutions for breadth first search, depth first graph search and A* search with h_ignore_preconditions.

Breadth First Search/A* search with h_ignore_preconditions:

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

Depth First Graph Search:

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

Table 2: Air Cargo Problem 2 Heuristics and Non-Heuristics Metrics

	Expansions	Goal Tests	New Nodes	Plan Length	Time(s)
Breadth First Search	3343	4609	30509	9	16.425
Depth First Graph Search	476	477	4253	466	2.684
Uniform Cost Search	4853	4855	44041	9	14.297
A* Search with h_ignore_preconditions	1450	1452	13303	9	5.009
A* Search with h_pg_levelsum	Took more then 10 minutes to run				

For problem 2, the non-heuristic results very similar to that of problem 1. Breadth first search, depth first graph search, and uniform cost search performed better in come categories, but not in all categories. For example, breadth first search and uniform cost search had the shortest path length but took longer to run search, while expanding more nodes. On the other hand, depth first search had the longest path length, but the shortest search time while expanding a fewer number of nodes. Based on this there is no clear optimal solution as each of the solutions have their advantages and disadvantages. The best optimal solution based on path length would be breadth first search as it had the shortest path length.

The A* search with heuristic function h_ignore_preconditons computed the solution in 5.009 seconds, with a path length of 9. But the heuristic function h_pg_levelsum didn't complete the search in under 10 minutes, so it had to be terminated. Because of this the best optimal heuristic function is h_ignore_preconditions. The solutions for both breadth first search and A* search with h_ignore_preconditions are provided below.

Breadth First Search:

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)

A* search with h_ignore_preconditions:

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)

Table 3: Air Cargo Problem 3 Heuristics and Non-Heuristics Metrics

	Expansions	Goal Tests	New Nodes	Plan Length	Time(s)
Breadth First Search	14663	18098	129631	12	118.098
Depth First Graph Search	1511	1512	12611	1442	15.166
Uniform Cost Search	18223	18225	159618	12	64.986
A* Search with h_ignore_preconditions	5040	5042	4494	12	20.006
A* Search with h_pg_levelsum	Took more than 10 minutes to run				

For problem 3, the results for non-heuristic searches were very similar to problems 1 and 2. The search with the shortest path length and fewest node expansions is breadth first search with a path length of 12. The search with the fastest time was depth first graph search, with a elapsed time of 15.166s. The search with a medium time and the shortest path length is uniform cost search.

The A* search with heuristic function h_ignore_preconditons computed the solution in 20.006 seconds, with a path length of 12. But the heuristic function h_pg_levelsum didn't complete the search in under 10 minutes, so it had to be terminated. Because of this the best optimal heuristic function is h_ignore_preconditions. The solutions for both uniform cost search and A* search with h_ignore_preconditions are provided below.

Uniform Cost Search:

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

A* search with h_ignore_preconditions:

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)

In conclusion, the best heuristic has been A* search with h_ignore_preconditons, as it's search time was the fastest for every problem. The heuristic function h_pg_levelsum didn't complete the search in under 10 ten except for problem 1. The heuristic searches were better overall for all problems compared to non-heuristics searches. Although on a few occasions the non-heuristics had a faster search times then heuristic searches, but their path lengths were

significantly longer. This shows that a very good heuristic function can make a big difference in finding the most optimal solution and not the just fastest solution or shortest solution.