Air Cargo Planning Problem

Heuristic Analysis - Sebastian Mack

The objective of this project was to solve a deterministic logistics planning problem for an Air Cargo transport system using a planning search agent. A basic problem description as well as concepts and applied algorithms can be found in [1]. Furthermore, all information regarding the detailed problem (including the action schema, initial states and goals) is available in the appendix.

Tables 1-3 provide the results gained for the AirCargo problems 1-3. Each table shows properties like plan length, optimality, time elapsed, number of node expansions for several search strategies that have been used. As was suggested in the Udacity instructions, some of the searches had to be skipped because the execution time exceeded 10 minutes.

Table 1 Results AirCargo Problem 1

Search Type	Plan Length	Optimal	Time [s]	Node Expansions
Breadth First Search	6	True	0.13	43
Breadth First Tree Search	6	True	5.08	1458
Depth First Graph Search	12	False	0.04	12
Depth Limited Search	50	False	0.47	101
Uniform Cost Search	6	True	0.2	55
Recursive Best First Search	6	True	15.3	4229
Greedy Best First Graph Search	6	True	0.03	7
A* Search h1 Heuristic	6	True	0.15	55
A* Search Ignore Preconditions Heuristic	6	True	0.14	41
A* Search Level Sum Heuristic	6	True	6.12	11

Table 2 Results AirCargo Problem 2

Search Type	Plan Length	Optimal	Time [s]	Node Expansions
Breadth First Search	9	True	36.63	3343
Breadth First Tree Search				
Depth First Graph Search	575	False	6.99	582
Depth Limited Search				
Uniform Cost Search	9	True	30.77	4852
Recursive Best First Search				
Greedy Best First Graph Search	17	False	7.11	990
A* Search h1 Heuristic	9	True	37.98	4852
A* Search Ignore Preconditions Heuristic	9	True	17.45	1450
A* Search Level Sum Heuristic	9	True	581.86	86

Table 3 Results AirCargo Problem 3

Search Type	Plan Length	Optimal	Time [s]	Node Expansions
Breadth First Search	12	True	296.14	14663
Breadth First Tree Search				
Depth First Graph Search	596	False	7.13	627
Depth Limited Search				
Uniform Cost Search	12	True	124.11	18235
Recursive Best First Search				
Greedy Best First Graph Search	22	False	50.03	5614
A* Search h1 Heuristic	12	True	128.33	18235
A* Search Ignore Preconditions Heuristic	12	True	74.96	5040
A* Search Level Sum Heuristic				

Optimal Plan for Problem 1

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Optimal Plan for Problem 2

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Load(C3, P3, ATL)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Optimal Plan for Problem 3

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(P2, ORD, SFO)

Fly(P1, ATL, JFK)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Non-heuristic Search Result Metrics

In this section, a comparison is given for the first seven rows of the tables 1-3 which represent the non-heuristic search strategies: Breadth First Search, Breadth First Tree Search, Depth First Graph Search, Depth Limited Search, Uniform Cost Search, Recursive Best First Search, Greedy Best First Graph. As mentioned before, not all of them could produce results in a reasonable time period for all three problems.

Only Breadth First Search and Uniform Cost Search could produce an optimal solution (true) for all three problems but the both had also a high memory consumption (node expansions) and a slow processing time (time elapsed).

Depth First Graph Search on the other hand, had excellent result for speed and memory usage but could not once produce an optimal result. An interesting option is the Greedy Best First Graph Search as it represents a good compromise between finding a solution with a tolerable path length and its processing time/memory consumption.

Heuristic Search Result Metrics

After analyzing non-heuristic search methods, this section compares their counterparts in form of the A* search algorithm with three different heuristic models. The respective results can be found in the rows 8-10 of tables 1-3 (A* Search h1 Heuristic, A* Search Ignore Preconditions Heuristic, A* Search Level Sum Heuristic).

All heuristic search strategies gained an optimal solution with exception of the A* Search Level Sum Heuristic that exceeded the 10 minutes time limit for problem 3. The A* Search h1 Heuristic required the highest memory storage in every case. In contrast, the A* Search Level Sum Heuristic had the smallest memory usage in 2/3 cases but also slowest processing time. A good compromise between the previous search types is the A* Search Ignore Preconditions Heuristic.

Summary

Comparing both heuristic and non-heuristic search methods, this section gives a short summary of the gained results. If an optimal solution is required heuristic methods were able to produce better results than non-heuristic methods. The best overall results have been produced by the A* Search Ignore Preconditions Heuristic which also performed well considering speed and memory usage. Non-heuristic methods like Breadth First Search could also calculate an optimal solution but showed far worse performance in speed and memory requirement. Depth First Graph Search performed excellent those categories but was not able to produce a tolerable plan length. Heuristic search stragies have proven their advantages in creating an optimal result with reasonable resources.

References

1. Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach **11,** 407-412 (2012).

Appendix

- 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:

```
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:

```
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:

```
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))
```