Air Cargo Planning Heuristic Analysis

Naruhiko Nakanishi

This project includes skeletons for the classes and functions needed to solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. With progression search algorithms like those in the navigation problem, optimal plans for each problem will be computed. Unlike the navigation problem, there is no simple distance heuristic to aid the agent. Instead, we implement domain-independent heuristics.

1. Classical PDDL problems

We define a group of problems in classical PDDL (Planning Domain Definition Language) for the air cargo domain. We set up the problems for search, experiment with various automatically generated heuristics, including planning graph heuristics, to solve the problems, and then provide an analysis of the results.

Problems 1, 2, and 3 are in the Air Cargo domain. All problems have the same action schema defined, but different initial states and goals as follows.

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))
```

Initial state and goal:

Problem 1	Problem 2	Problem 3
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))	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))	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. Optimal Sequence of Actions

An optimal sequence of actions is identified for each problem. The optimal plan lengths for Problems 1, 2, and 3 are 6, 9, and 12 actions, respectively.

Optimal plan:

Problem 1	Problem 2	Problem 3
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
Fly(P1, SFO, JFK)	Fly(P1, SFO, JFK)	Fly(P2, JFK, ORD)
Unload(C1, P1, JFK)	Load(C2, P2, JFK)	Load(C4, P2, ORD)
Load(C2, P2, JFK)	Fly(P2, JFK, SFO)	Fly(P2, ORD, SFO)
Fly(P2, JFK, SFO)	Load(C3, P3, ATL)	Unload(C4, P2, SFO)
Unload(C2, P2, SFO)	Fly(P3, ATL, SFO)	Load(C1, P1, SFO)
	Unload(C3, P3, SFO)	Fly(P1, SFO, ATL)
	Unload(C2, P2, SFO)	Load(C3, P1, ATL)
	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
		Unload(C3, P1, JFK)
		Unload(C2, P2, SFO)
		Unload(C1, P1, JFK)

3. Search Strategies Analysis

The following tables show the performance of the algorithms on the problems compared, including the optimality of the solutions (Yes: if a solution of optimal length is found, No: otherwise), time elapsed (in terms of speed), and the number of node expansions required (in terms of memory usage).

Problem 1

	optimality	Time elapsed (sec)	node expansions	Plan length
breadth_first_search	Yes	0.032	43	6
depth_first_graph_search	No	0.015	21	20
uniform_cost_search	Yes	0.039	55	6
astar_search with h_1	Yes	0.043	55	6
astar_search with h_ignore_preconditions	Yes	0.039	41	6
astar_search with h_pg_levelsum	Yes	1.205	11	6

Problem 2

	optimality	Time elapsed (sec)	node expansions	Plan length
breadth_first_search	Yes	14.487	3343	9
depth_first_graph_search	No	3.703	624	619
uniform_cost_search	Yes	12.657	4853	9
astar_search with h_1	Yes	12.299	4853	9
astar_search with h_ignore_preconditions	Yes	4.555	1450	9
astar_search with h_pg_levelsum	Yes	190.824	86	9

Problem 3

	optimality	Time elapsed (sec)	node expansions	Plan length
breadth_first_search	Yes	110.970	14663	12
depth_first_graph_search	No	1.883	408	392
uniform_cost_search	Yes	54.421	18223	12
astar_search with h_1	Yes	54.701	18223	12
astar_search with h_ignore_preconditions	Yes	17.373	5040	12
astar_search with h_pg_levelsum	Yes	1012.516	325	12

3.1 Uninformed Search Strategies Analysis

We compare and contrast non-heuristic search result metrics, which are optimality, time elapsed, number of node expansions, for Problems 1, 2, and 3. Breadth-first, depth-first, and uniform-cost are

included in our comparison.

Depth first graph search is the fastest and uses the least memory, but it is not optimal. The plan lengths for Problems 1, 2, and 3 are 20, 619, and 392, respectively. An optimal solution is defined that it has the lowest path cost among all solutions [1]. Depth first graph search is not optimal because it does not consider if a node is better than another. It simply explores the nodes that take it as deep as possible in the graph even if the goal node is to its right [1].

Breadth first search and uniform cost search are optimal. Among them, breadth first search is a little bit faster for Problem 1, however, uniform cost search is faster for Problems 2, and 3. This means that uniform cost search does better performance in terms of time as the complexity of the problems increase. On the other hand, breadth first search uses less memory than uniform cost search for Problems 1, 2, and 3.

3.2 Informed (Heuristic) Search Strategies Analysis

We compare and contrast heuristic search result metrics using A* with the "h_ignore preconditions" and "h_pg_level-sum" and “h_1” for Problems 1, 2, and 3.

All three heuristic search strategies above are optimal. The plan lengths for Problems 1, 2, and 3 are 6, 9, and 12, respectively. Among them, A* with the "h_ignore preconditions" is the fastest even if A* with the "h_pg_level-sum" uses the least memory. There is the trade-off between speed and memory usage.

4. Discussion

Among informed and Uninformed Search Strategies that are optimal, breadth first search is the fastest for Problem 1. However, A* with the "h_ignore preconditions" is the fastest for Problems 2, and 3. This means that A* with the "h_ignore preconditions" does better performance in terms of time as the complexity of the problems increase.

A* with the "h_ignore preconditions" heuristic was the best heuristic used in these problems. However, it was not better in terms time than breadth first search which is non-heuristic search planning method for problem 1. This might be because heuristic is too complex for the simple problem like Problem 1.

References

1. Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition)