Heuristic Analysis for Air Cargo Planning Agent

By Joseph Mejorada

Introduction

In this project, a planning agent was developed to solve air cargo transportation planning problems. The problems solved in this project represent classical PDDL (Planning Domain Definition Language) problems. The results were gathered using uninformed and heuristic search methods. Performance of these methods was compared based on the path length of the search and the time required to solve for a solution.

Problem Definition

Three problems regarding air cargo transportation were analyzed using three uninformed search methods and three methods that use heuristics. All problems use the same action schema (see **Figure 1**), but each has their own initial state and goal (see **Figure 2**).

```
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))
```

Figure 1: Air Cargo Action Schema.

- 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))
```

Figure 2: Initial state and goal for each Air Cargo problem.

Test Results

The uninformed search methods used were breath first search, depth first graph search, and uniform cost search. For the heuristic searches, three variations for A* search was used: h1, h_ignore_preconditions, and h_level_sum. Performance comparisons of these search methods solving for the three air cargo problems are depicted in **Figure 3**. The optimality score was determined by calculating the fastest method with the least number of actions to reach the given goal. The higher the score, the more optimal the method is. **Figure 4** demonstrates the path steps of the most optimal search method for each problem.

Problem	Test	# Expansions	# of Goal Tests	New Nodes	Plan Length	Time Elapsed	Optimality Score
1	breadth_first_search	43	56	180	6	0.02559937	5
	depth_first_graph_search	21	22	84	20	0.01412293	n/a
	uniform_cost_search	55	57	224	6	0.0570936	2
	astar_search h_1	55	57	224	6	0.0308378	4
	astar_search h_ignore_preconditions	41	43	170	6	0.04121186	3
	astar_search h_pg_levelsum	11	13	50	6	1.24194972	1
2	breadth_first_search	3343	4609	30509	9	13.4292481	2
	depth_first_graph_search	624	625	5602	619	3.43862797	n/a
	uniform_cost_search	4853	4855	44041	9	10.1388392	4
	astar_search h_1	4853	4855	44041	9	10.6248388	3
	astar_search h_ignore_preconditions	1450	1452	13303	9	3.78574154	5
	astar_search h_pg_levelsum	86	88	841	9	205.207221	1
3	breadth_first_search	14663	18098	129631	12	100.07362	2
	depth_first_graph_search	408	409	3364	393	1.8849892	n/a
	uniform_cost_search	18222	18224	159608	12	44.5158371	4
	astar_search h_1	18222	18224	159608	12	45.82351	3
	astar_search h_ignore_preconditions	5042	5042	44944	12	14.4919356	5
	astar_search h_pg_levelsum	316	318	2912	12	1194.22514	1

Figure 3: Results outputted by script 'run_search.py'. Methods highlighted in green represent the most optimal method for the given problem.

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

Figure 4: The shortest path to acheive each problem's goal was acheived using Breadth First Search for Problem #1 and A* Search while ignoring preconditions for Problems #2 and #3.

Recommendation

Based on the results depicted in **Figure 3**, depth first graph search proved to be an unfeasible method to use when solving planning problems. While this method was the quickest way to find a solution, the method resulted in a large number of steps. For simple problems such as Problem #1, breadth first graph search seems to be the most optimal method. Breadth first search always considers the shortest path first, and with simple problems, will find a solution in a relatively short amount of time [1].

However, the processing speed of breadth first graph search slows down as the problems become more complex (such as in Problem #2 and #3). As complexity increases, A* search with the heuristic of ignoring preconditions proves to be the best search method. This method searches for the solution with the least number of steps to reach its goal while searching for the solution at a quicker speed by removing preconditions from actions.

References

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