Artificial Intelligence Nanodegree

Implementing a Planning Search

Guillermo Aure
Aug 05, 2017

Introduction

The following analysis contains benchmarking information about a Cargo air transportation problems; it consists of finding the less possible number of actions to move cargo between airports using planes. The information here represents the data obtained from running different search and graph search algorithms to find an optimal solution to such problem.

The analysis contains a table with the result of running multiple search algorithms. Some of the algorithms use heuristics others do not. When an algorithm search is using heuristics, it gets specified in the table.

Air Cargo Transportation Problems Definitions

What follows is the definitions of three Air Cargo Transportation problems. Each problem is expressed using "Planning Domain Definition Language."

For simplification I have ommited the negative effects of each initial state.

In the below representation the letters state the following:

- C = Cargo
- P = Plane
- SFO = San Francisco International Airport
- JFK = John F. Kennedy International Airport
- ATL = HartsfieldJackson Atlanta International Airport
- ORD = O'Hare International Airport

- **Problem 1**:

  cargos = ['C1', 'C2']
  planes = ['P1', 'P2']
  airports = ['JFK', 'SFO']

  Initial State = ['At(C1, SFO)','At(C2, JFK)','At(P1, SFO)','At(P2, JFK)']
  The goal = ['At(C1, JFK)','At(C2, SFO)']

- **Problem 2**:

  cargos = [’C1’, ’C2’, ’C3’]
  planes = [’P1’, ’P2’, ’P3’]
  airports = [’JFK’, ’SFO’, ’ATL’]

  Initial State = [’At(C1, SFO)’, ’At(C2, JFK)’, ’At(C3, ATL)’, ’At(P1, SFO)’, ’At(P2, JFK)’, ’At(P3, ATL)’]
  The goal = [’At(C1, JFK)’, ’At(C2, SFO)’, ’At(C3,SFO)’]

- **Problem 3**:
  cargos = [’C1’, ’C2’, ’C3’, ’C4’]
  planes = [’P1’, ’P2’]
  airports = [’JFK’, ’SFO’, ’ATL’, ’ORD’]

  Initial State = [’At(C1, SFO)’, ’At(C2, JFK)’, ’At(C3, ATL)’, ’At(C4, ORD)’, ’At(P1, SFO)’, ’At(P2, JFK)’]
  The goal = [’At(C1, JFK)’, ’At(C2, SFO)’, ’At(C3,JFK)’, ’At(C4,SFO)’]

Optimal Plans

Follow the optimal plans for each of the three problems defined in the previous section.

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)
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(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P3, SFO)

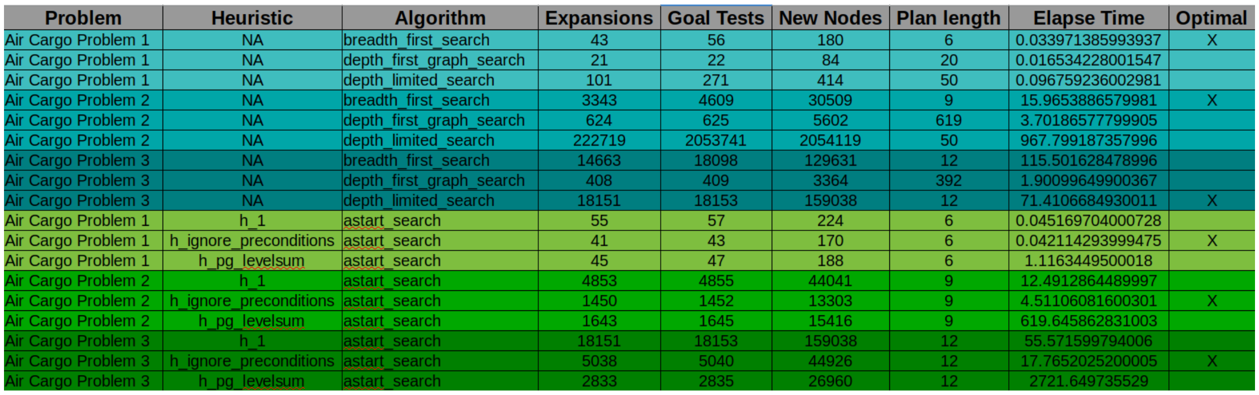
Problem 3:

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(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P1, JFK) Unload(C4, P2, SFO)

Non-Heuristic Result Metric Analysis

After executing three different non heuristic search algorithms to find a solution of the problems previously state we conclude the following.

The three uninformed search methods used were:
Breadth First Search
Depth First Graph Search
Depth Limited Search



Problem	Heuristic	Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Elapse Time	Optimal
Air Cargo Problem 1	NA	breadth_first_search	43	56	180	6	0.033971385993937	X
Air Cargo Problem 1	NA	depth_first_graph_search	21	22	84	20	0.016534228001547	
Air Cargo Problem 1	NA	depth_limited_search	101	271	414	50	0.096759236002981	
Air Cargo Problem 2	NA	breadth_first_search	3343	4609	30509	9	15.9653886579981	X
Air Cargo Problem 2	NA	depth_first_graph_search	624	625	5602	619	3.70186577799905	
Air Cargo Problem 2	NA	depth_limited_search	222719	2053741	2054119	50	967.799187357996	
Air Cargo Problem 3	NA	breadth_first_search	14663	18098	129631	12	115.501628478996	
Air Cargo Problem 3	NA	depth_first_graph_search	408	409	3364	392	1.90099649900367	
Air Cargo Problem 3	NA	depth_limited_search	18151	18153	159038	12	71.4106684930011	X
Air Cargo Problem 1	h_1	astart_search	55	57	224	6	0.045169704000728	
Air Cargo Problem 1	h_ignore_preconditions	astart_search	41	43	170	6	0.042114293999475	X
Air Cargo Problem 1	h_pg_levelsum	astart_search	45	47	188	6	1.1163449500018	
Air Cargo Problem 2	h_1	astart_search	4853	4855	44041	9	12.4912864489997	
Air Cargo Problem 2	h_ignore_preconditions	astart_search	1450	1452	13303	9	4.51106081600301	X
Air Cargo Problem 2	h_pg_levelsum	astart_search	1643	1645	15416	9	619.645862831003	
Air Cargo Problem 3	h_1	astart_search	18151	18153	159038	12	55.571599794006	
Air Cargo Problem 3	h_ignore_preconditions	astart_search	5038	5040	44926	12	17.7652025200005	X
Air Cargo Problem 3	h_pg_levelsum	astart_search	2833	2835	26960	12	2721.649735529	

Figure 1: Results Table

In the results table (Figure 1), we can appreciate that except the problem 3 ”Depth Limited Search,” the only algorithm that produced a complete and optimal solution was the ”Breadth First Search.” Regarding time complexity and space complexity the ”Breadth First Search” is not the best search strategy nevertheless, it guarantees to find an optimal solution if one exists and the branching factor is finite. Contrary to ”Breadth First Search” the other two search strategies did not find an optimal solution (except on problem 3); these were faster than the ”Breadth First Search” and in the case of the ”Depth First Graph Search” it even expanded fewer nodes, but it returned a non-optimal solution. The ”Depth Limited Search” exception on problem three returned an optimal solution. The ”Depth Limited Search” can be complete or incomplete depending on the problem solution depth (Section 3.4.4 of Artificial Intelligence Modern Approach Book.) We know based on the results of a complete optimal search like ”Breadth First Search,” that the depth of the solution for problem 3 is 12 steps. Now for ”Depth Limit Search” with a ”limit” less than 12 we would obtain an incomplete solution. In this experiment that is not the case; the ”limit” is set to 50, which guarantees that at least a solution will be found if one exists. Now if the ”limit” is set to a value much greater than 12 the solution can become non-optimal. Examples of this scenario are problems one and two where a limit of ”50” produces nonoptimal plans because their solutions depth is way less than ”50” steps away. For a problem with a solution depth of 12, a limit of 50 seems to produce an optimal plan which is the case of the exception we described at the beginning of this paragraph for the problem 3.

Heuristic Result Metric Analysis

After executing three different heuristic search algorithms to find a solution of the problems previously state we conclude the following.

The two heuristic search I used were:
Ignore Preconditions
PG Levelsum

Looking at the Figure 1 table result, we can note that no matter which of the two heuristics we used the A* algorithm always find an optimal solution. Same as in the non-heuristic search methods space (expanded nodes) and time also varies but this time because of the heuristic function instead of the search process. In the "Ignore Pre-Condition" heuristic function we measure how close a state is to achieve the goal of a problem by checking the state literals and subtracting from it the number of goal literals. The closest to achieve the goal the more goal literals such state should content. The PG Levelsum does something similar, but instead of counting based on states, it counts base on graph levels. The use of a heuristic function improves the completeness and optimality of a search method.

Of the two heuristics "Ignore Preconditions" is the best heuristic; both have a similar estimated effective branching factor (Section 3.6.1 of Artificial Intelligence Modern Approach Book), but the "Ignore Preconditions" find a solution faster than the "PG Levelsum"

We used the below formula
(http://ozark.hendrix.edu/ ferrer/courses/335/f11/lectures/effective-branching.html) to estimate the branching factor of both heuristics in Problem two.

N(̂1/d)

Where "N" is the number of expanded nodes, and "d" is the solution depth. Assuming that the plan length is equal to the solution depth, the effective branching factors of the two heuristics mention before for problem two are:

Ignore Preconditions = 2.24
PG Levesum = 2.27

The reason why "Ignore Preconditions" is a faster heuristic, is because the "PG levelsum" heuristic has to calculate the level cost of each goal and add them up, this is computational more difficult than the relaxation strategy of the "Ignore Preconditions" heuristic (Section 10.2.3 "Decomposition" and Section 10.3.1 "Level Sum" of Artificial Intelligence Modern Approach Book.)