AI Planning Project

Kirill Romanov

February 27th , 2017

Air cargo problem. Heuristic report

Artificial Intelligence Nanodegree

Summary

1. Truly informed planning searches algorithms (*astar_search with h_ignore_preconditions* and *astar_search with h_pg_levelsum*)[1] provide optimal solution in all considered problems and use for it less number of node expansions
2. The best results in term of optimality and time demonstrate *astar_search with h_ignore_ preconditions* algorithm
3. However, *astar_search with h_pg_levelsum* algorithm also provides optimal solution and is best in term of number of node expansions. Unfortunately, the computational time of heuristic is big and as a result the total computational time is considerable.
4. Between uninformed algorithms, **optimal solution** in reasonable time **can provide breadth_first_search** (best in this class in term of optimality, time and new nodes quantity) and *uniform_cost_search algoritm/Astar_with_h_1* (optimal, but less efficient than BFS) .
5. The algorithms which are not provided optimal solution seem to be impractical due to huge gap between proposed and optimal solution
6. We should note that the most reasonable cause of fail of some algorithms based on h_1 heuristic it is heuristic, not the algorithms itself.[2] It would be perspective to test these algorithms with more meaning heuristics.

Problem 1. Result analysis

1. The first problem is very small and simple, so we can test each algorithm and define it applicability for the problem space before to test it on problems that are more complex.
2. The results of tests are in the table below:

#	Algoritm Name	Time	Length	Expansion	Goal Test	New Nodes
1	breadth_first_search	0.15	6	43	56	180
2	**breadth_first_tree_search**	4.8	6	1458	1459	5960
3	depth_first_graph_search	0.05	12	12	13	48
4	**depth_limited_search**	0.37	50	101	271	414
5	uniform_cost_search	0.2	6	55	57	224
6	**recursive_best_first_search with h_1**	14.9	6	4229	4230	17029
7	**greedy_best_first_graph_search with h_1**	0.02	6	7	9	28
8	astar_search with h_1	0.18	6	55	57	224
9	**astar_search with h_ignore_preconditions**	0.12	6	31	33	126
10	**astar_search with h_pg_levelsum**	2.8	6	11	13	50

Table 1. Problem 1 results

[1] We cannot consider astar_search with h_1 as a truly informed algoritm because h_1 heuristic doesn't provide any useful information. In fact the results of this algorithm in term of optimality and nodes are equal to uniform_cost_search. The time is a little bit slower due to additional computations resources to add constant. Due to this we will not consider this algorithm in our analysis.

[2] For example, RBFS is described by Peter Norvig as "RBFS (recursive best-first search) and SMA∗ (simplified memory-bounded A∗) are robust, optimal search algorithms that use limited amounts of memory; given enough time, they can solve problems that A∗ cannot solve because it runs out of memory." See "Artificial intelligence a modern approach" 3rd edition, section 3.7 "Summary" for more information

3. From these results we can made the following conclusions:
    a. Breadth_first_tree_search is impractical for usage. Usage of tree search instead of grid search means unnecessary repeating of already done steps[3] Due to this it is impossible to use it for problems that are more complex[4].
    b. recursive_best_first_search with h_1 cannot run efficiently without reasonable heuristic. As h_1 is not reasonable heuristic, number of nodes expanded is huge and it failed to search more complex problem
    c. depth_limited_search provides unreasonable solution (50 moves instead of possible 6 is not relevant solution) and number of expanded nodes is big.

Problem 2 and 3. Result analysis

1. To understand the efficiency of algorithms on more complex problems it is useful to analyze problem 2 and problem 3 together. Below are the results of computations[5]

#	Algoritm Name	Time	Length	Expansion	Goal Test	New Nodes
1	breadth_first_search	66.7	9	3343	4609	30509
2	breadth_first_tree_search	cannot be computed within reasonable time				
3	depth_first_graph_search	13.7	575	582	583	5211
4	depth_limited_search	cannot be computed within reasonable time				
5	uniform_cost_search	147.9	9	4852	4854	44030
6	recursive_best_first_search with h_1	cannot be computed within reasonable time				
7	greedy_best_first_graph_search with h_1	31.6	15	990	992	8910
8	astar_search with h_1	159.8	9	4852	4854	44030
9	**astar_search with h_ignore_preconditions**	39.5	9	1166	1168	10651
10	astar_search with h_pg_levelsum	265	9	86	88	841

Table 2. Problem 2 results

#	Algoritm Name	Time	Length	Expansion	Goal Test	New Nodes
1	breadth_first_search	461.5	12	14663	18098	129631
2	breadth_first_tree_search	cannot be computed within reasonable time				
3	depth_first_graph_search	17.3	596	627	628	5176
4	depth_limited_search	cannot be computed within reasonable time				
5	uniform_cost_search	1009.4	12	18235	18237	159716
6	recursive_best_first_search with h_1	cannot be computed within reasonable time				
7	greedy_best_first_graph_search with h_1	294.8	22	5615	5617	49438
8	astar_search with h_1	1143.6	12	18235	18237	159716
9	**astar_search with h_ignore_preconditions**	205	12	3816	3818	34216
10	astar_search with h_pg_levelsum	1724.6	12	404	406	3718

Table 3. Problem 2 results

2. As we can see, when the problem increase, non-optimal algorithms became impractical (for example, greedy_best_first_graph_search with h_1 became two time less optimal during problem 3 and consequently it is non reasonable to use it for practical problems)

[3] For more information see “Artificial intelligence a modern approach” 3rd edition, section 3.3 “Searching for solution”
[4] To prove this state for this algorithms and for other, I try to run search for problem 2 and 3, but it failed to compute it in reasonable time
[5] If the algorithm name font is green, it is best algorithm to solve the problem. If is red – unacceptable. If is black – not best, but can be used for the problem

3. Though, breadth_first_search algoritm is acceptable for problem 3, it is obvious if the problem became more complex it is failed to solve within reasonable time due to state complexity (and uniform_cost_search already fail to pass 10 minutes limit)
4. Finally, as we can see in the results above, *ignore_preconditions* heuristic can reduce path to solve as much as 3-5 times, comparing with uninformed search, and provide reasonable time for computation. As a result **astar_search with h_ignore_preconditions** is the winner in our test
5. h_pg_levelsum heuristic is the best in term of path search (it is 10 time less comparing with *ignore_preconditions* heuristic). However it is hard to compute and as a result, total calculation time of algorithm with this heuristic is very slow and it is became unpractical to use it on complex problem (problem 3 and more complex problems)

Optimal solution review

1. In general case the following algorithms let find the optimal sequence:
   a. breadth_first_search
   b. uniform_cost_search/ astar_search with h_1
   c. astar_search with h_ignore_preconditions
   d. astar_search with h_pg_levelsum
2. However, in the simplest problem (problem 1), optimal solution was find by greedy best first search (not the case for more complex problems)
3. In tables below are optimal sequences which were identified for each problem:

Problem	Optimal sequence

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	Optimal sequence

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