Udacity Project 2 – Building a Forward Planning Agent Analysis

By Sanuja Cooray

I finished implementing the forward planning agent for the air cargo problems using uninformed search methods such as depth first, breadth first searches as well as informed heuristic-based methods such as A* search.

AIR CARGO PROBLEM 1

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Plant Length	Time Elapsed
breadth_first_search	20	43	56	178	6	0.00325
depth_first_graph_search	20	21	22	84	20	0.18468
uniform_cost_search	20	60	62	240	6	0.00719
greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.00081
greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0.30027
greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0.23199
greedy_best_first_graph_search with h_pg_setlevel	20	45	47	190	8	2.87976
astar_search with h_unmet_goals	20	50	52	206	6	0.00498
astar_search with h_pg_levelsum	20	28	30	122	6	0.75060
astar_search with h_pg_maxlevel	20	43	45	180	6	0.76971
astar_search with h_pg_setlevel	20	55	57	226	6	2.97981

AIR CARGO PROBLEM 2

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Plant Length	Time Elapsed
breadth_first_search	72	3343	4609	30503	9	3.20010
depth_first_graph_search	72	624	625	5601	619	2.19591
uniform_cost_search	72	5154	5156	46618	9	2.62415
greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.01499
greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	8.81525
greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	17.44322
greedy_best_first_graph_search with h_pg_setlevel	72	1417	1419	13755	16	2,803.14510
astar_search with h_unmet_goals	72	2467	2469	22522	9	1.77349
astar_search with h_pg_levelsum	72	357	359	3426	9	219.05384
astar_search with h_pg_maxlevel	72	2887	2889	26594	9	1,283.13405
astar_search with h_pg_setlevel	72	4367	4369	39865	9	7,581.21999

Due to the higher plan length which requires substantially more memory and have very long execution times, I decided to exclude the following searches for the air cargo problems 3 and 4:

- *uniform_cost_search*
- *depth_first_search*
- *astar_search with h_pg_maxlevel*
- *astar_search with h_pg_setlevel*

AIR CARGO PROBLEM 3

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Plant Length	Time Elapsed
breadth_first_search	88	14663	18098	129625	12	8.27497
depth_first_graph_search	88					
uniform_cost_search	88					
greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0.02836
greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	21.14804
greedy_best_first_graph_search with h_pg_maxlevel	88	21	23	195	13	25.50003
greedy_best_first_graph_search with h_pg_setlevel	88	11102	11104	101690	18	24,901.93310
astar_search with h_unmet_goals	88	7388	3790	65711	12	6.63636
astar_search with h_pg_levelsum	88	369	371	3403	12	383.42775
astar_search with h_pg_maxlevel	88					
astar_search with h_pg_setlevel	88					

AIR CARGO PROBLEM 4

Search Algorithm	Actions	Expansions	Goal Tests	New Nodes	Plant Length	Time Elapsed
breadth_first_search	104	99736	114953	944130	14	74.53859
depth_first_graph_search	104					
uniform_cost_search	104					
greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	18	0.04909
greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	39.17369
greedy_best_first_graph_search with h_pg_maxlevel	104	56	58	580	17	93.76718
greedy_best_first_graph_search with h_pg_setlevel	104					
astar_search with h_unmet_goals	104	34330	34332	328509	14	43.65275
astar_search with h_pg_levelsum	104	1208	1210	12210	15	2,199.29806
astar_search with h_pg_maxlevel	104					
astar_search with h_pg_setlevel	104					

ANALYZING SEARCH COMPLEXITY

As the complexity of the problem increases, the search complexity rises across all algorithms however it does not rise equally across all algorithms. Algorithms such as depth_first_search see an exponential increase in search complexity as the plan length increases dramatically in comparison to other uninformed searches.

However, in other uninformed searches, the number of nodes explored, and the expansions made rises dramatically as the problem complexity increases. The same phenomena is observed in the A* search algorithms as well where the search complexity increases dramatically, this similar relationship could be due to the shared attributes of breadth_first_search and A* searches even though the A* search has the additional layer of using a heuristic as well.

Fascinatingly, the algorithm that is purely heuristic based, namely the greedy_best_first_search performs the best with the least increase in search complexity as the problem complexity increases.

Considering the heuristics used across both A* and greedy_best_first_search, it is clear that h_pg_levelsum and h_pg_setlevel showcase the least increase search complexity. I believe this is due to the increased accuracy of the heuristics in related to the problem.

ANALYZING TIME

As the complexity of the problem increases, the time taken to solve the problem increases across all algorithms however it does not increase equally across all the algorithms and all heuristics.

When considering the algorithms, greedy_best_first_search once again performed the best with the least increase in time and least time taken to complete the solution.

Algorithms that used heuristics such as h_pg_setlevel and h_pg_maxlevel observed an exponential increase in time taken whereas in relation, heuristics such h_unmet_goals saw the least increase. I believe, h_unmet_goals performed the best because other heuristics simply take too long to calculate, although the accuracy of those heuristics maybe better, the time take to calculate the heuristics simply makes them infeasible as the complexity of the problem rises.

ANALYZING OPTIMALITY OF THE SOLUTION

I would personally base the optimality of the solution simply based on resource utilization specifically memory consumed which is linked to increased search complexity and most importantly time taken.

In both cases the pure heuristic based method of greedy_best_first_search comes out on top with the least increase in search complexity hence least increase in memory utilization while also maintaining the least increase in time.

When evaluating the heuristics, h_pg_levelsum and h_pg_setlevel have the least increase in search complexity with increasing problem complexity however I do not see them as the ideal solution simply due to the exponential increase in time consumed due to the complexity of calculating the heuristic. Since I prioritize time consumed over memory consumed, I would say the heuristic of h_umet_goals is fair better due to the substantially less increase in time consumed.

QUESTIONS

1) Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

   Considering that real time applications require real time feedback, I would prioritize execution time to be the most important hence for such a situation I would use the greedy_best_first_search algorithm with h_unmet_goals heuristic.

2) Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)?

   Since the time restriction is a lower priority here, I would use A* search combined with a good heuristic function. I would still pick the h_unmet_goals function here simply due to the complexity of the problem, the other heuristics will simply take far too long to calculate thereby making infeasible.

3) Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

   Since the most optimal plans are the final goal and if time consumed and memory utilized are not important. I would simply use and uninformed search method specifically breath_first_search or uniform_cost_search to find the most optimal plan as these algorithms are guaranteed to converge on the most optimal plans due to thorough search of the problem space in comparison to other algorithms which maybe be biased with the use of heuristics and end up with suboptimal plans. However, as the problem complexity rises the requirement of resources to solve this problem also exponentially rises.