Project 2: Classical Planning

Miguel Tasende

May 28, 2018

1 Introduction

This report shows the results of solving some "Air Cargo" (logistics planning) problems with different search algorithms (on top of a graph plan).

2 Problems 1 and 2

The first two problems were solved using each of the search algorithms. The results can be seen on the table below.

problem		1				2			
	search_algorithm	actions	expansions	time_s	path_length	actions	expansions	time_s	path_length
index									
1	breadth_first_search	20	43	0.021	6	72	3343	0.383	9
2	depth_first_graph_search	20	21	0.007	20	72	624	0.551	619
3	uniform_cost_search	20	60	0.022	6	72	5154	0.628	9
4	greedy_best_first_graph_search with h_unmet_goals	20	7	0.002	6	72	17	0.013	9
5	greedy_best_first_graph_search with h_pg_levelsum	20	6	0.57	6	72	9	2.401	9
6	greedy_best_first_graph_search with h_pg_maxlevel	20	6	0.18	6	72	27	2.976	9
7	greedy_best_first_graph_search with h_pg_setlevel	20	6	0.615	6	72	27	9.611	9
8	astar_search with h_unmet_goals	20	50	0.018	6	72	2467	0.776	9
9	astar_search with h_pg_levelsum	20	28	0.388	6	72	357	49.888	9
10	astar_search with h_pg_maxlevel	20	43	0.308	6	72	2887	267.257	9
11	astar_search with h_pg_setlevel	20	42	0.776	6	72	1987	680.743	9

Figure 1: Results for the first two problems, with all the search algorithms

It was noticed that the Depth First algorithm was resulting in extremely long paths, and considering this is an Air Cargo problem, it is reasonable to expect that optimum or near-optimum solutions are searched for. Therefore, the Depth First algorithm won't be considered in the rest of the project.
A^* with "maxlevel" and "setlevel" heuristics are taking a lot of time to find a solution, but they will be considered for the other problems, to have more data (it is expected that running all the remaining combinations may take about 10 hours of processing with pypy3).

3 Problems 3 and 4

Problems 3 and 4 were solved with all the algorithms except Depth First. In the case of Problem 4, the A* algorithm with the "project goals set level" heuristic didin't finish in

a reasonable time, and so it was stopped. The results are below.

problem		3	3	3	3	4	4	4	4
	search_algorithm	actions	expansions	time_s	path_length	actions	expansions	time_s	path_length
index									
1	breadth_first_search	88	14663	1.017	12	104	99736	5.844	14
3	uniform_cost_search	88	18510	1.872	12	104	113339	9.024	14
4	greedy_best_first_graph_search with h_unmet_goals	88	25	0.056	15	104	29	0.059	18
5	greedy_best_first_graph_search with h_pg_levelsum	88	14	4.782	14	104	17	7.231	17
6	greedy_best_first_graph_search with h_pg_maxlevel	88	21	5.218	13	104	56	15.444	17
7	greedy_best_first_graph_search with h_pg_setlevel	88	41	33.04	17	104	128	139.781	23
8	astar_search with h_unmet_goals	88	7388	1.991	12	104	34330	4.904	14
9	astar_search with h_pg_levelsum	88	369	80.091	12	104	1208	446.462	15
10	astar_search with h_pg_maxlevel	88	9580	1403.757	12	104	62077	15148.963	14
11	astar_search with h_pg_setlevel	88	6092	3795.374	12				

Figure 2: Results for the last two problems

4 Aggregated Results

Some tables and charts are shown below, aggregating the results for all problems, and all search algorithms in different ways.

4.1 Number of expanded nodes

Actions	20	72	88	104
algorithm				
greedy_best_first_graph_search with h_pg_levelsum	6	9	14	17
greedy_best_first_graph_search with h_pg_maxlevel	6	27	21	56
greedy_best_first_graph_search with h_unmet_goals	7	17	25	29
greedy_best_first_graph_search with h_pg_setlevel	6	27	41	128
astar_search with h_pg_levelsum	28	357	369	1208
astar_search with h_pg_setlevel	42	1987	6092	
astar_search with h_unmet_goals	50	2467	7388	34330
astar_search with h_pg_maxlevel	43	2887	9580	62077
breadth_first_search	43	3343	14663	99736
uniform_cost_search	60	5154	18510	113339
depth_first_graph_search	21	624		

Figure 3: Table showing the number of expanded nodes against the number of actions for each algorithm.

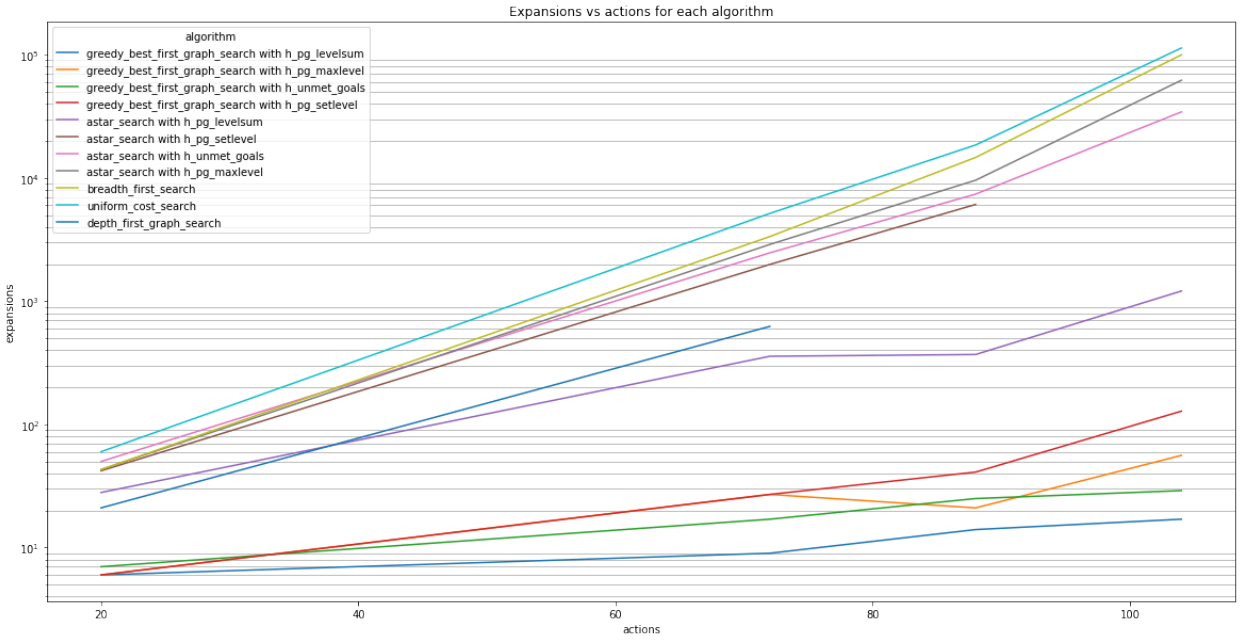

Figure 4: Chart showing the number of expanded nodes against the number of actions for each algorithm. The y-axis scale is logarithmic.

The growth of the expanded nodes is exponential ($Ce^{\alpha a}$) for all the algorithms, but each one has a different α and C.

4.2 Time to complete the search

Actions	20	72	88	104
algorithm				
greedy_best_first_graph_search with h_unmet_goals	0.002	0.013	0.056	0.059
breadth_first_search	0.021	0.383	1.017	5.844
uniform_cost_search	0.022	0.628	1.872	9.024
astar_search with h_unmet_goals	0.018	0.776	1.991	4.904
greedy_best_first_graph_search with h_pg_levelsum	0.57	2.401	4.782	7.231
greedy_best_first_graph_search with h_pg_maxlevel	0.18	2.976	5.218	15.444
greedy_best_first_graph_search with h_pg_setlevel	0.615	9.611	33.04	139.781
astar_search with h_pg_levelsum	0.388	49.888	80.091	446.462
astar_search with h_pg_maxlevel	0.308	267.257	1403.757	15148.963
astar_search with h_pg_setlevel	0.776	680.743	3795.374	
depth_first_graph_search	0.007	0.551		

Figure 5: Table showing the time to complete the search (in seconds) against the number of actions for each algorithm.

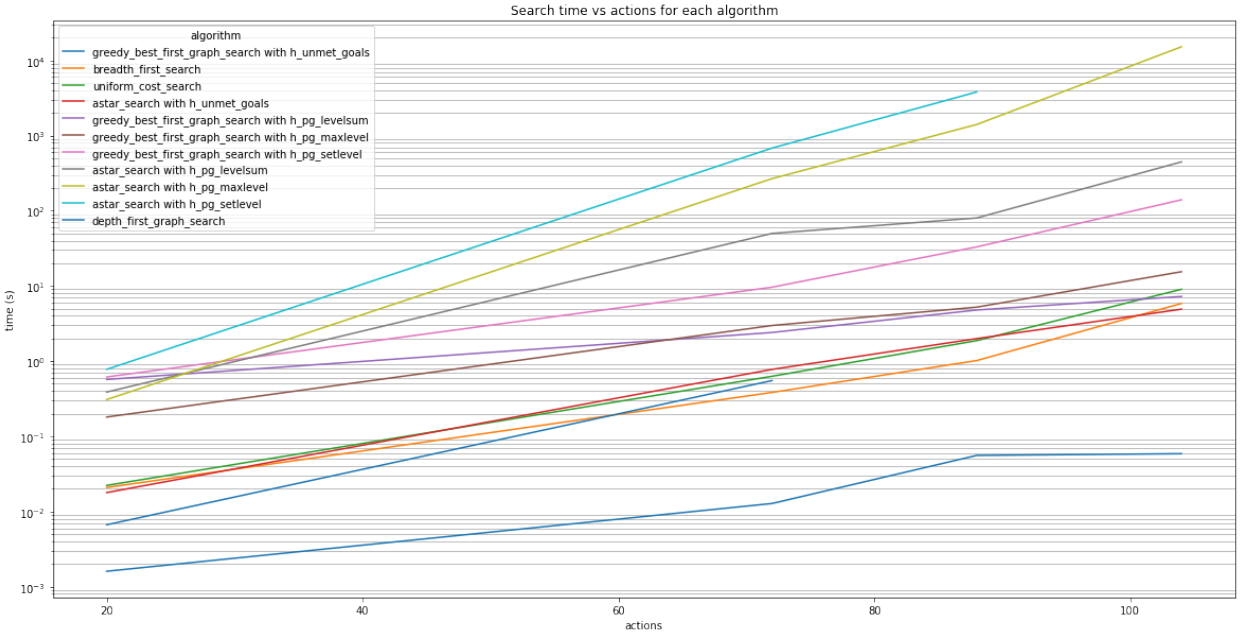

Figure 6: Chart showing the time to complete the search (in seconds) against the number of actions for each algorithm. The y-axis scale is logarithmic.

The growth of the search time is exponential ($Ce^{\alpha a}$) for all the algorithms, but each one has a different α and C.

4.3 Length of the plan for each problem

Problem	1	2	3	4
algorithm				
astar_search with h_pg_levelsum	6	9	12	15
astar_search with h_pg_maxlevel	6	9	12	14
astar_search with h_pg_setlevel	6	9	12	
astar_search with h_unmet_goals	6	9	12	14
breadth_first_search	6	9	12	14
uniform_cost_search	6	9	12	14
greedy_best_first_graph_search with h_pg_maxlevel	6	9	13	17
greedy_best_first_graph_search with h_pg_levelsum	6	9	14	17
greedy_best_first_graph_search with h_unmet_goals	6	9	15	18
greedy_best_first_graph_search with h_pg_setlevel	6	9	17	23
depth_first_graph_search	20	619		

Figure 7: Length of the resulting plan for each problem and each algorithm.

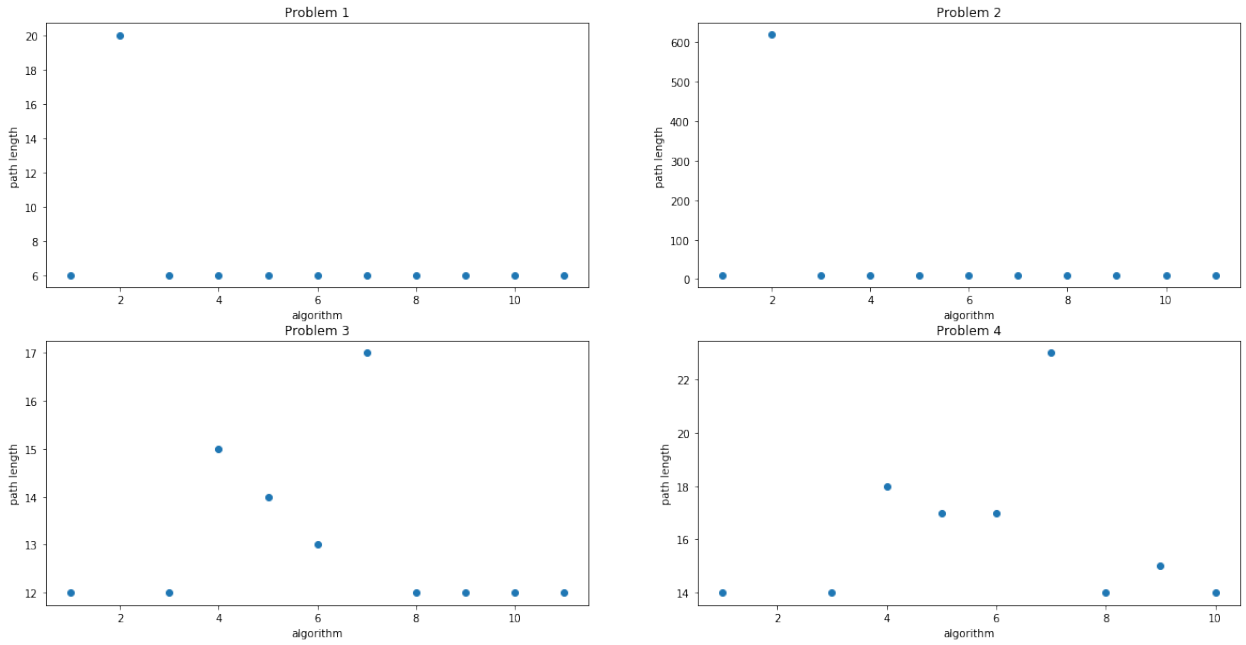

Figure 8: Length of the resulting plan for each problem and each algorithm.

It can be readily seen that many of the algorithms aren't optimal. Except the case of Depth First Search, the others are "not so far away" from the optimal solution. The algorithm ids references can be seen below (the x-axis was too small to use the entire names):

1	breadth_first_search
2	depth_first_graph_search
3	uniform_cost_search
4	greedy_best_first_graph_search with h_unmet_goals
5	greedy_best_first_graph_search with h_pg_levelsum
6	greedy_best_first_graph_search with h_pg_maxlevel
7	greedy_best_first_graph_search with h_pg_setlevel
8	astar_search with h_unmet_goals
9	astar_search with h_pg_levelsum
10	astar_search with h_pg_maxlevel
11	astar_search with h_pg_setlevel

Figure 9: Algorithm ids references.

5 Results Analysis

5.1 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?

The answer would depend on the precise meaning of "real time" and "a few actions". Assuming that a sub-second response is desired and that there are about 20 to 70 actions, I would choose Breadth First Search, because it guarantees optimality, it can comply with the time restrictions, and it is very simple. If the "real time" condition is more restrictive, one could use Greedy Best First Graph Search with the "unmet goals" heuristic. That won't guarantee the optimality but it will be extremely fast (tens of milliseconds, less than the blink of an eye [that takes about 250ms]). Uniform Cost Search would be acceptable in some cases, but it offers no advantage to Breadth First Search.

5.2 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)

A complete cost-benefit analysis may reveal that it may be worth dedicating more computing resources, or software optimization efforts to getting better results, but I will answer considering my laptop performance with the current algorithms... In that case, if there are some tens of thousands of UPS drivers, the only algorithm that seems to be able to cope with the problem, in a reasonable time [less than 12 hours, assuming the planning is done in the night before, for example], is Greedy Best First Graph Search with the "unmet goals" heuristic. It doesn't guarantee optimality of the path, but the experimental lengths in the assignment were not so bad. The second choice would be A* with the "unmet goals" heuristic (doesn't guarantee optimality, and takes longer time, but may get closer-to-optimal results, in some cases)

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

Breadth First Search, Uniform Cost Search, A* with the "problem goals maxlevel" heuristic, or A* with the "problem goals setlevel" heuristic. None of the greedy algorithms can

guarantee optimality of the path, nor can Depth First Search. The "unmet goals" and "level sum" heuristics are not admissible, and so they can't be used if optimality is needed.