Project 2 Forward Planning Agent

Introduction

This project's purpose is to build a Forward Planning Agent to solve 4 cargo problems. These problems have different complexity with respect to number of airplanes, cargo items and airports. The result is that their numbers of actions in the domain will range from 20 to 104.

To solve the cargo problems, there are 11 search algorithms being tested in this project. Three of them are uninformed search algorithms: Breadth First Search, Depth First Search and Uniform Cost Search. The other eight are heuristics algorithms. Four of the heuristics are Greedy Best First graph search and the other four are A* search. These heuristics are tested with different cost functions: unmet goals, level sum, max level and set level.

To make it easy to follow, we encode these 11 search algorithms with numbers as below.

Table 1: Search algorithm reference

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

Below is the complexity in term of number of actions for the four cargo problems.

Table 2: Number of actions in cargo problems

Cargo Problem	Number of actions
1	20
2	72
3	88
4	104

Experiments

After updating the code in the project, I ran all eleven search algorithms for the cargo problems 1 and 2. All of the algorithms ran within 2 seconds for the problem 1. For problem 2, the A* search algorithms for Level Sum, Max Level and Set Level had significant slower run time than the others. Especially for A* with Set Level, the run time was 1649 seconds or almost half an hour.

Because of long runtime, I didn't run A* with Set Level for problem 3 and didn't run both A* with max level and A* with set level for problem 4.

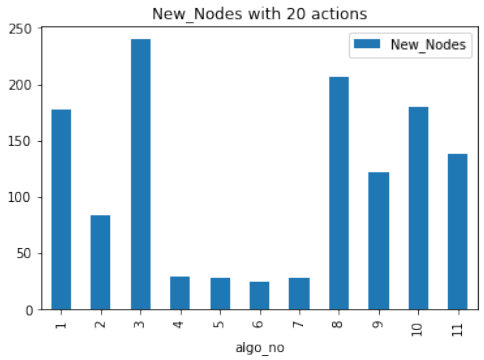
Another initial observation is that Depth First Search creates very long plans for all problems so it is not a good approach for these problems.

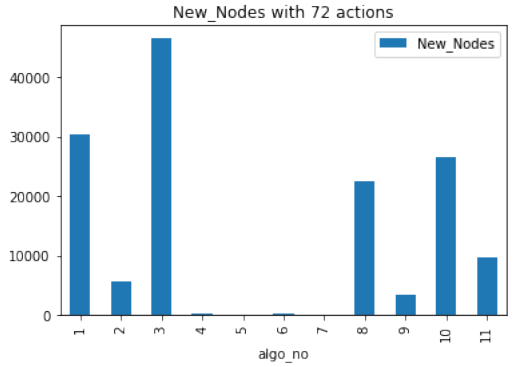
The performance of these algorithms is analyzed in term of number of nodes, search time and plan length as below.

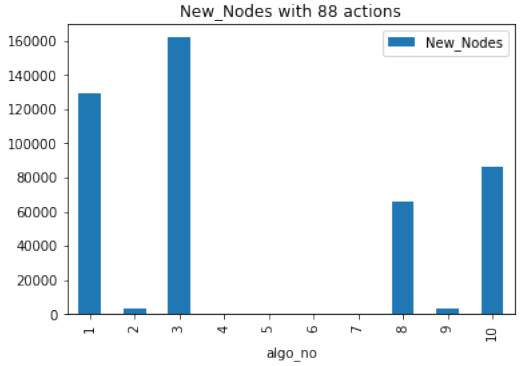
Results

Number of nodes

As we can see, breadth first search and uniform cost search have a greater number of new nodes than the others. On the contrary, all of the greedy best first search algorithms have very low number of new nodes. The reason is that greedy best first search only uses the estimated distance to the target for the next node in the frontier without checking thoroughly all of the other nodes.







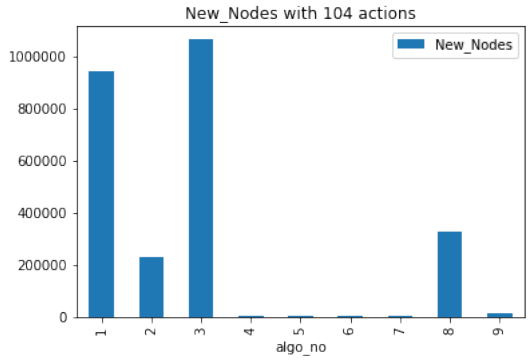


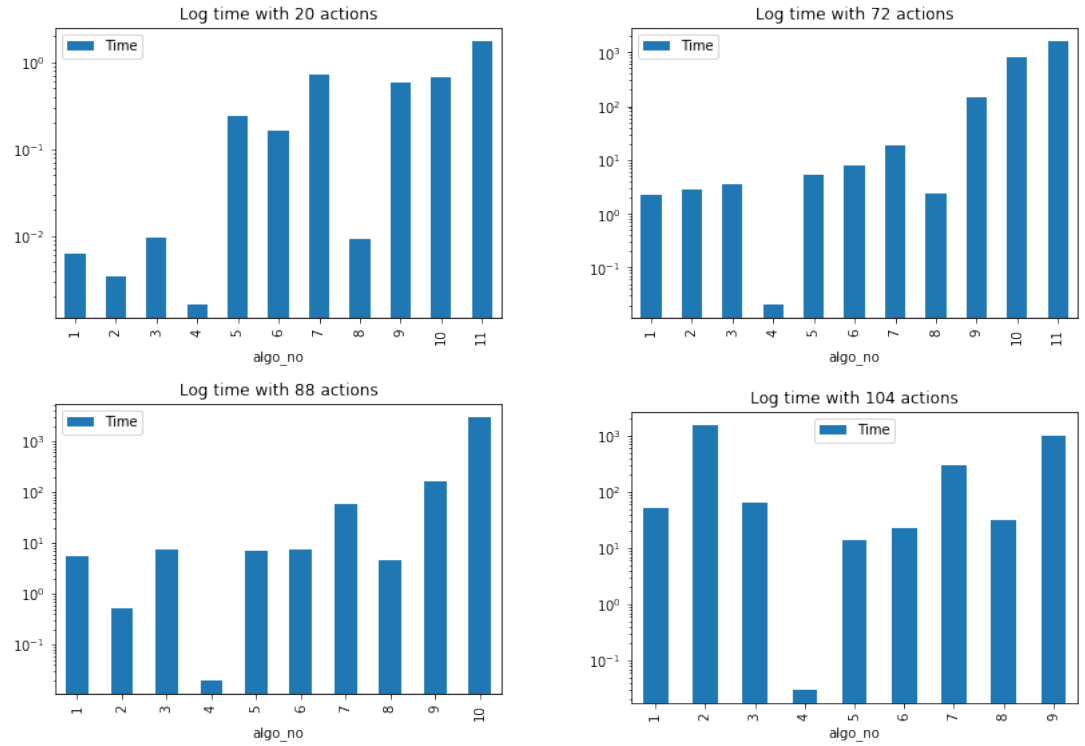
Table 3: Number of nodes vs search algorithms and action numbers

	Search Algorithms										
Action Numbers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
20	178	84	240	29	28	24	28	206	122	180	138
72	30,503	5,602	46,618	170	86	249	84	22,522	3,426	26,594	9,605
88	129,625	3,364	161,936	230	126	195	345	65,711	3,403	86,312	
104	944,130	228,849	1,066,413	280	165	580	1,164	328,509	12,210		

Run Time

Table 4: Search time vs search algorithms and action numbers

Action Numbers	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***
20	0.01	0.00	0.01	0.00	0.24	0.16	0.73	0.01	0.58	0.67	1.78
72	2.20	2.84	3.53	0.02	5.31	8.12	18.50	2.31	143.68	824.03	1,649.06
88	5.66	0.52	7.63	0.02	7.20	7.25	59.70	4.45	161.81	3,023.30	x
104	51.65	1,551.46	65.69	0.03	13.81	23.35	295.95	31.26	984.30	x	x



For the small problem (20 actions), the differences between the algorithms' run times are not significant. A* with set level is slowest with 1.78 seconds. All of the uninformed search algorithms and heuristics with unmet goals have run time of 0 to 0.01 second.

For bigger problems, we start to see the big differences between the algorithms' run times. In general, A* Search with Max Level and Set Level have significantly longer run times than the others. For 88-action problem, A* with Max Level finished in 50 minutes and A* with Set Level would have run for even longer time if I had run it. For 104-action problem, I didn't run A* with Max Level and A* with Set Level. For this very complicated problem, A* with Sum Level finished in 16 minutes and A* with Unmet Goals finished in 30 seconds.

Tables 5 and 6 show us the effect of different level cost functions on run times for Greedy Best First Search and A* Search. We can see that using these functions for estimating level cost really takes a toll in run times for problems with large action sets. Furthermore, since A* Search seeks to search more nodes than Greedy Best First Search, the effect of the cost function on A* Search algorithms becomes even more significant.

Table 5: Effect of different level cost functions to run times for Greedy Best First Search

	Run times				Ratio to Unmet goal			
Problem	**Unmet goal**	**Sum level**	**Max level**	**Set level**	**Unmet goal**	**Sum level**	**Max level**	**Set level**
1	0.00	0.24	0.16	0.73	1.00	148.55	99.79	454.57
2	0.02	5.31	8.12	18.50	1.00	261.46	399.65	910.52
3	0.02	7.20	7.25	59.70	1.00	372.64	374.98	3,087.94
4	0.03	13.81	23.35	295.95	1.00	449.82	760.48	9,638.85

Table 6: Effect of different level cost functions to run times for A Search*

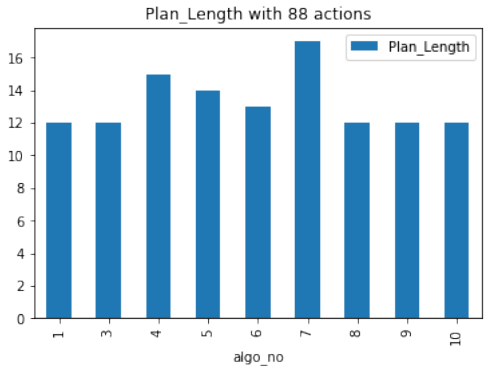
	Run times				Ratio to Unmet goal			
Problem	**Unmet goal**	**Sum level**	**Max level**	**Set level**	**Unmet goal**	**Sum level**	**Max level**	**Set level**
1	0.01	0.58	0.67	1.78	1.00	62.27	72.05	191.39
2	2.31	143.68	824.03	1,649.06	1.00	62.25	357.01	714.46
3	4.45	161.81	3,023.30	x	1.00	36.37	679.55	-
4	31.26	984.30	x	x	1.00	31.49	-	-

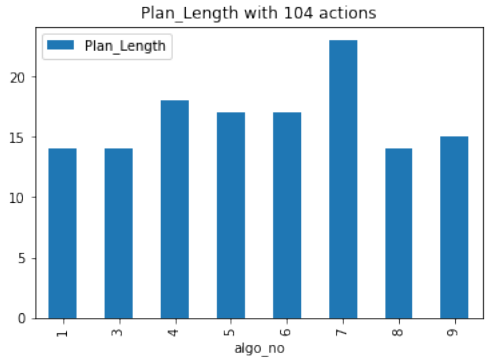
Plan Length

As we can see from the table, Depth First Search consistently finds very long plans for all of the problems. All of the other algorithms don't differ too much from each other for the length of the plans to reach the goals. However, it seems like Greedy Best First Search will give slightly longer plans for bigger problems. The graphs below show plan length for problem 3 and 4 excluding Depth First Search.

Table 7: Plan length vs search algorithms and action numbers

Action Numbers	1	2	3	4	5	6	7	8	9	10	11
20	6	20	6	6	6	6	6	6	6	6	6
72	9	619	9	9	9	9	9	9	9	9	9
88	12	392	12	15	14	13	17	12	12	12	x
104	14	24132	14	18	17	17	23	14	15	x	x





Project Questions

Q1: 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?

Answer: An algorithm should be quick to operate in real time. However, for restricted domain, all of the algorithms perform pretty well. Except for A* Search with Set Level which ran for almost 2 seconds, all of the other algorithms finish within 1 second. However, since the uninformed algorithms might require more memory because they explore more nodes, it might be better to use the informed heuristics. The A* Search algorithms can give optimal results so I think it is better to use any of the A* Search algorithms. If there's a constraint of run time then we will choose the one that can produce results less than that limit.

Q2: 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)

Answer: For very large domain, run time really matters. Although we don't need to find the solution in a couple of seconds, we cannot run the search for hours to get the best plan. In my opinion, the algorithms below are all good candidates for this problem since their run times for our biggest problem (104 actions) are less than or a bit over one minute.

- Breadth First Search,
- Uniform Cost Search,

- Greedy Best First Search with Unmet Goal,
- Greedy Best First Search with Level Sum and
- Greedy Best First Search with Max Level,
- A* Search with Unmet Goal

However, since our action set is significantly large (based on UPS fact sheet, their fleet has over 5,900 vehicles) and run time increases exponentially with action numbers, we might have to use Greedy Best First Search with Unmet Goal for this problem because it always yields very quick run time.

Note: UPS fact sheet link:
https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=FactSheets&id=1426321563187-193

Q3: Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Answer: Of all of the algorithms, Breadth First Search and A* Search are optimal. Breadth First Search has a much greater number of new nodes than A* Search but its run time is always shorter than the A* Search algorithms with level cost functions (Max Level, Sum Level and Set Level). Meanwhile Breadth First Search run time is longer than A* Search with Unmet Goal.

For smaller problems, we can use A* Search with Level Sum and Set Level because they have fewer nodes than Breadth First Search and the other A* Search. However, for larger problems, it's better to use Breadth First Search or A* Search with Unmet Goal. A* Search with Unmet Goal is more advantageous than Breadth First Search because it has shorter runtime and fewer nodes.