Classical Planning Report

January 13, 2019

1 Classical Planning Exploration

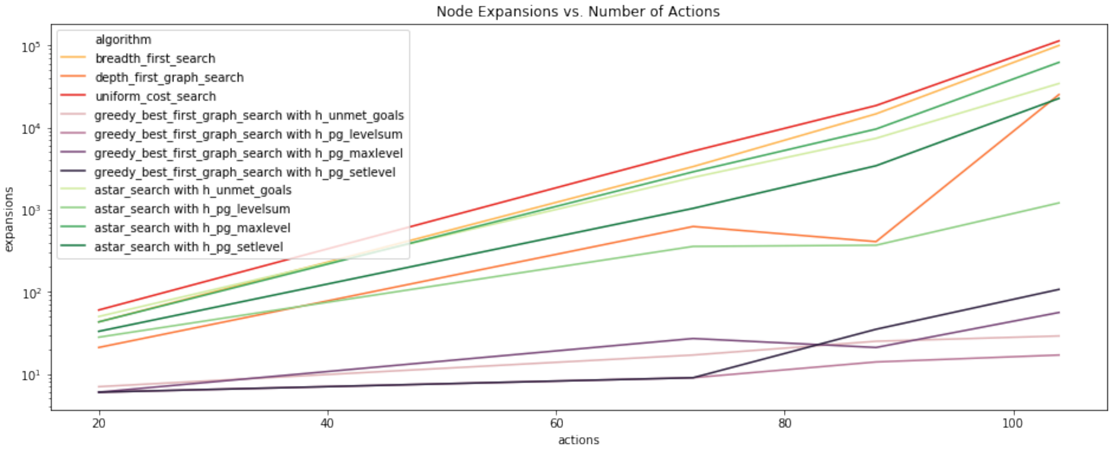
The complete table of results comparing all eleven search algorithms for each of the four planning problems.

```
    problem                                   algorithm  actions  \
0         1                        breadth_first_search       20
1         1                   depth_first_graph_search       20
2         1                         uniform_cost_search       20
3         1  greedy_best_first_graph_search with h_unmet_goals       20
4         1  greedy_best_first_graph_search with h_pg_levelsum       20
5         1  greedy_best_first_graph_search with h_pg_maxlevel       20
6         1  greedy_best_first_graph_search with h_pg_setlevel       20
7         1               astar_search with h_unmet_goals       20
8         1               astar_search with h_pg_levelsum       20
9         1               astar_search with h_pg_maxlevel       20
10        1               astar_search with h_pg_setlevel       20
11        2                        breadth_first_search       72
12        2                   depth_first_graph_search       72
13        2                         uniform_cost_search       72
14        2  greedy_best_first_graph_search with h_unmet_goals       72
15        2  greedy_best_first_graph_search with h_pg_levelsum       72
16        2  greedy_best_first_graph_search with h_pg_maxlevel       72
17        2  greedy_best_first_graph_search with h_pg_setlevel       72
18        2               astar_search with h_unmet_goals       72
19        2               astar_search with h_pg_levelsum       72
20        2               astar_search with h_pg_maxlevel       72
21        2               astar_search with h_pg_setlevel       72
22        3                        breadth_first_search       88
23        3                   depth_first_graph_search       88
24        3                         uniform_cost_search       88
25        3  greedy_best_first_graph_search with h_unmet_goals       88
26        3  greedy_best_first_graph_search with h_pg_levelsum       88
27        3  greedy_best_first_graph_search with h_pg_maxlevel       88
28        3  greedy_best_first_graph_search with h_pg_setlevel       88
29        3               astar_search with h_unmet_goals       88
30        3               astar_search with h_pg_levelsum       88
```

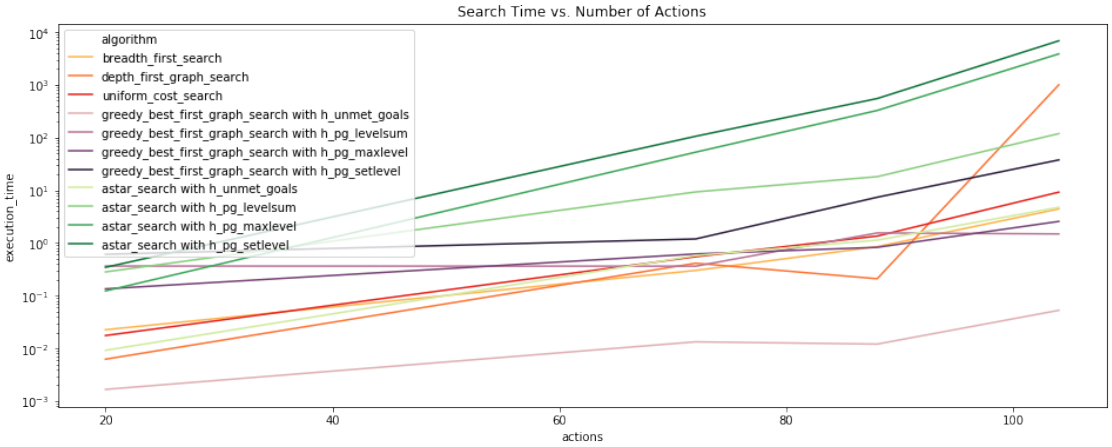
```
31        3                    astar_search with h_pg_maxlevel       88
32        3                    astar_search with h_pg_setlevel       88
33        4                               breadth_first_search      104
34        4                           depth_first_graph_search      104
35        4                                uniform_cost_search      104
36        4  greedy_best_first_graph_search with h_unmet_goals      104
37        4  greedy_best_first_graph_search with h_pg_levelsum      104
38        4  greedy_best_first_graph_search with h_pg_maxlevel      104
39        4  greedy_best_first_graph_search with h_pg_setlevel      104
40        4                    astar_search with h_unmet_goals      104
41        4                    astar_search with h_pg_levelsum      104
42        4                    astar_search with h_pg_maxlevel      104
43        4                    astar_search with h_pg_setlevel      104

    expansions  goal_tests  new_nodes  plan_length  execution_time
0           43          56        178            6        0.022632
1           21          22         84           20        0.006242
2           60          62        240            6        0.017548
3            7           9         29            6        0.001667
4            6           8         28            6        0.364773
5            6           8         24            6        0.134647
6            6           8         28            6        0.609376
7           50          52        206            6        0.009193
8           28          30        122            6        0.282280
9           43          45        180            6        0.123078
10          33          35        138            6        0.345047
11        3343        4609      30503            9        0.301007
12         624         625       5602          619        0.412987
13        5154        5156      46618            9        0.548182
14          17          19        170            9        0.013318
15           9          11         86            9        0.363038
16          27          29        249            9        0.619586
17           9          11         84            9        1.191494
18        2467        2469      22522            9        0.577189
19         357         359       3426            9        9.294133
20        2887        2889      26594            9       52.780004
21        1037        1039       9605            9      105.682291
22       14663       18098     129625           12        0.851343
23         408         409       3364          392        0.209368
24       18510       18512     161936           12        1.351013
25          25          27        230           15        0.012072
26          14          16        126           14        1.549254
27          21          23        195           13        0.826525
28          35          37        345           17        7.368583
29        7388        7390      65711           12        1.131607
30         369         371       3403           12       18.096431
31        9580        9582      86312           12      326.427136
32        3423        3425      31596           12      549.598873
```

```
33        99736       114953      944130           14       4.419489
34        25174        25175      228849        24132     999.968523
35       113339       113341     1066413           14       9.215409
36           29           31         280           18       0.052743
37           17           19         165           17       1.485746
38           56           58         580           17       2.564387
39          107          109        1164           23      37.625936
40        34330        34332      328509           14       4.743122
41         1208         1210       12210           15     118.650650
42        62077        62079      599376           14    3863.493351
43        22606        22608      224229           14    6845.869624
```

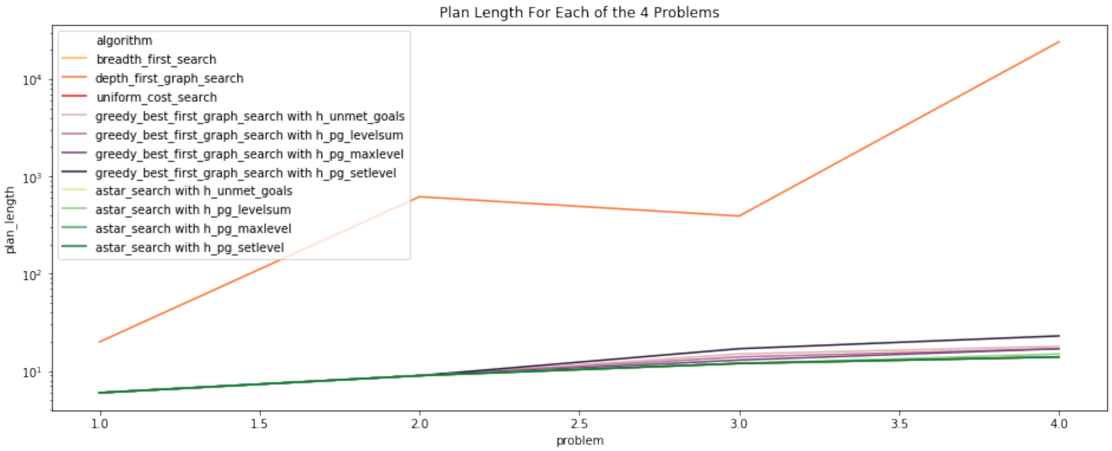
1.1 Use a table or chart to analyze the number of nodes expanded against number of actions in the domain

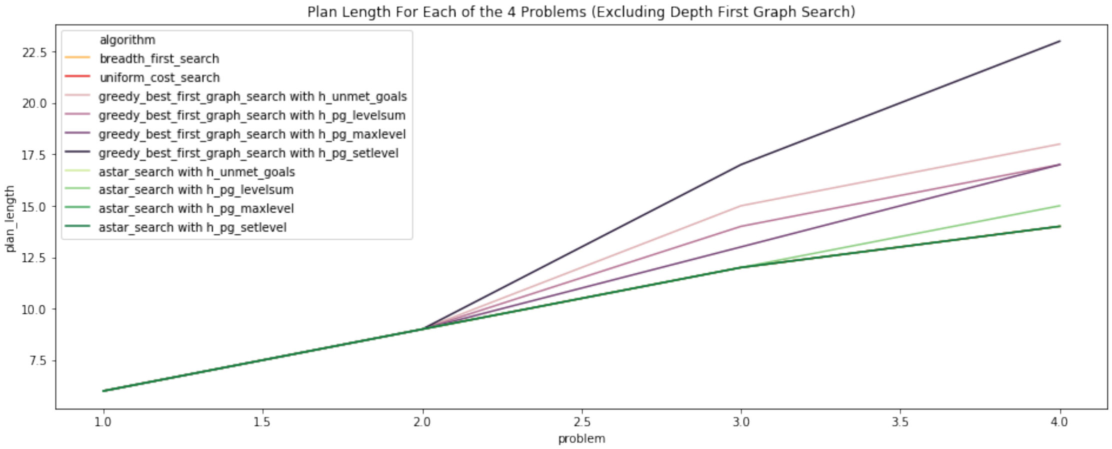


1.2 Use a table or chart to analyze the search time against the number of actions in the domain



1.3 Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems





1.4 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?

For problems that have few actions, all of the algorithms perform equally well in terms of finding the optimal plan. They don't, however, perfrom equally well in terms of the nodes expanded and how long the search took to execute. The most important consideration when dealing with real time applications is the time it takes for the search to execute. With this in mind, the **greedy best first graph search with the unmet goals heuristic** executed 3.7 times faster than the next fastest execution for problem 1 and 365 times faster than the algorithm with the slowest execution.

1.5 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)

It depends what is most important to optimise for the problem at hand. If node expansion (memory / space requirements) and execution time are most important when searching for a plan that solves a complex problem, then **greedy best first graph search** expands the fewest nodes and searching with the **unmet goals heuristic** it executes the fastest. However, these algorithms performed the worst when it came to finding the optimal path. I would argue for a problems such as logistics planning where taking unnecessary steps to complete a plan would cost a company far more than the search execution time. In this case, finding the optimal plan is more important which suggests that **A*** would be a better choice and executing it using the **unmet goals heuristic** minimises the execution time and node expansion while optimising the plan length.

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

Without a doubt the most successful search algorithm when it came to consistently finding the optimal plan is **A***. All of the algorithms found the optimal plan for problems 1 and 2 but the performance of the algorithms began to diverge for problems 3 and 4. All of heuristics used with **A*** found the optimal plan for problem 3 while all but one found the optimal plan for problem 4. The **unmet goals heuristic** provides the best balance between performance and search execution time while consistently finding the optimal path.