Build a Forward-Planning Agent – Report

Collection of data from Experiments

In the following sections are included the tables from the data gathered from running the experiments with the different solutions, all solutions from (1 to 11) for problems 1 and 2. Then, a selection of solutions is made to be applied to problems 3 and 4 based on the results for problems 1/2.

Data from Problem 1

#	Solution	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time Elapsed (seconds)
1	breadth_first_search	20	43	56	178	6	0.006207066
2	depth_first_graph_search	20	21	22	84	20	0.003602394
3	uniform_cost_search	20	60	62	240	6	0.009746118
4	greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.001590145
5	greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0.467545498
6	greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0.350782159
7	greedy_best_first_graph_search with h_pg_setlevel	20	6	8	28	6	0.640830027
8	astar_search with h_unmet_goals	20	50	52	206	6	0.009646022
9	astar_search with h_pg_levelsum	20	28	30	122	6	1.204123405
10	astar_search with h_pg_maxlevel	20	43	45	180	6	1.246777534
11	astar_search with h_pg_setlevel	20	33	35	138	6	1.509046707

Data from Problem 2

#	Solution	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time Elapsed (seconds)
1	breadth_first_search	72	3343	4609	30503	9	1.992882182
2	depth_first_graph_search	72	624	625	5602	619	3.139584953
3	uniform_cost_search	72	5154	5156	46618	9	3.544576849

4	greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.019723868
5	greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	10.844764705
6	greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	21.828859935
7	greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	9	15.743980093
8	astar_search with h_unmet_goals	72	2467	2469	22522	9	2.211288832
9	astar_search with h_pg_levelsum	72	357	359	3426	9	270.063894836
10	astar_search with h_pg_maxlevel	72	2887	2889	26594	9	1570.252354011
11	astar_search with h_pg_setlevel	72	1037	1039	9605	9	1420.532363647

Analysis from Problems 1 and 2

In general it can be observed that the amount of expanded nodes increases with the number of actions. The uninformed methods (with the exception of DFS) are the ones that produce the most of node expansions followed by the methods based on A*. The methods which by far produce the least of the expansions are the ones based on greedy-best-first-search, the number of expansion these produce don’t reach the levels of hundreds or thousands as the other families of methods. It’s also worth to mention that in general the *h_pg_levelsum* heuristic has a significant impact on reducing the number of expansions.

In terms of the length of paths, all the methods except DFS give the same lengths. It’s expected by the nature of DFS that the lengths of the paths generated for it would be much grater.

In regards to the search time, in general the uninformed methods provide the fastest execution time, followed closely for the greedy-best-first-search, and then a lot more slower the A* based methods, which executions in problem 2 take several minutes. Also it’s worth to mention that the *h_unmet_goals* heuristic has a very significant impact on reducing the execution time.

Based on this analysis, considering mainly execution time, secondarily number of expansion and finally length of paths, the chosen methods to be tested on problems 3 and 4 are:

- breadth_first_search
- uniform_cost_search
- greedy_best_first_graph_search with h_unmet_goals
- greedy_best_first_graph_search with h_pg_levelsum
- astar_search with h_unmet_goals
- astar_search with h_pg_levelsum

Data from Problem 3

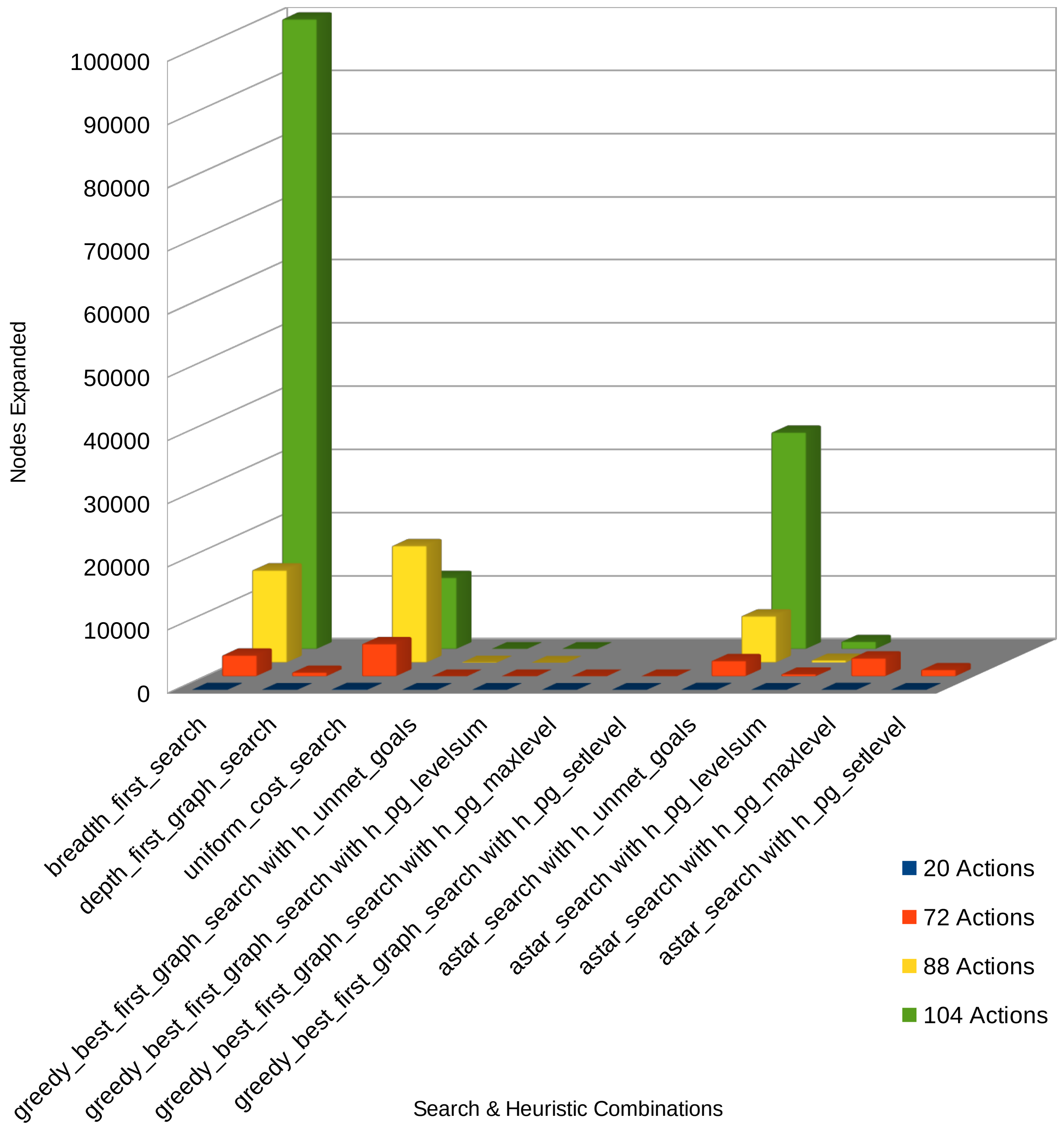
#	Solution	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time Elapsed (seconds)
1	breadth_first_search	88	14663	18098	129625	12	10.858450656
3	uniform_cost_search	88	18510	18512	161936	12	14.388250982
4	greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0.036721673
5	greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	23.508493250
8	astar_search with h_unmet_goals	88	7388	7390	65711	12	8.399828981
9	astar_search with h_pg_levelsum	88	369	371	3403	12	427.797650590

Data from Problem 4

#	Solution	Actions	Expansions	Goal Tests	New Nodes	Path Length	Time Elapsed (seconds)
1	breadth_first_search	104	99736	114953	944130	14	95.819584726
3	uniform_cost_search	104	113339	113341	1066413	14	122.325290053
4	greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	18	0.086775484
5	greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	45.590738482
8	astar_search with h_unmet_goals	104	34330	34332	328509	14	101.887208339
9	astar_search with h_pg_levelsum	104	1208	1210	12210	15	2387.696166984

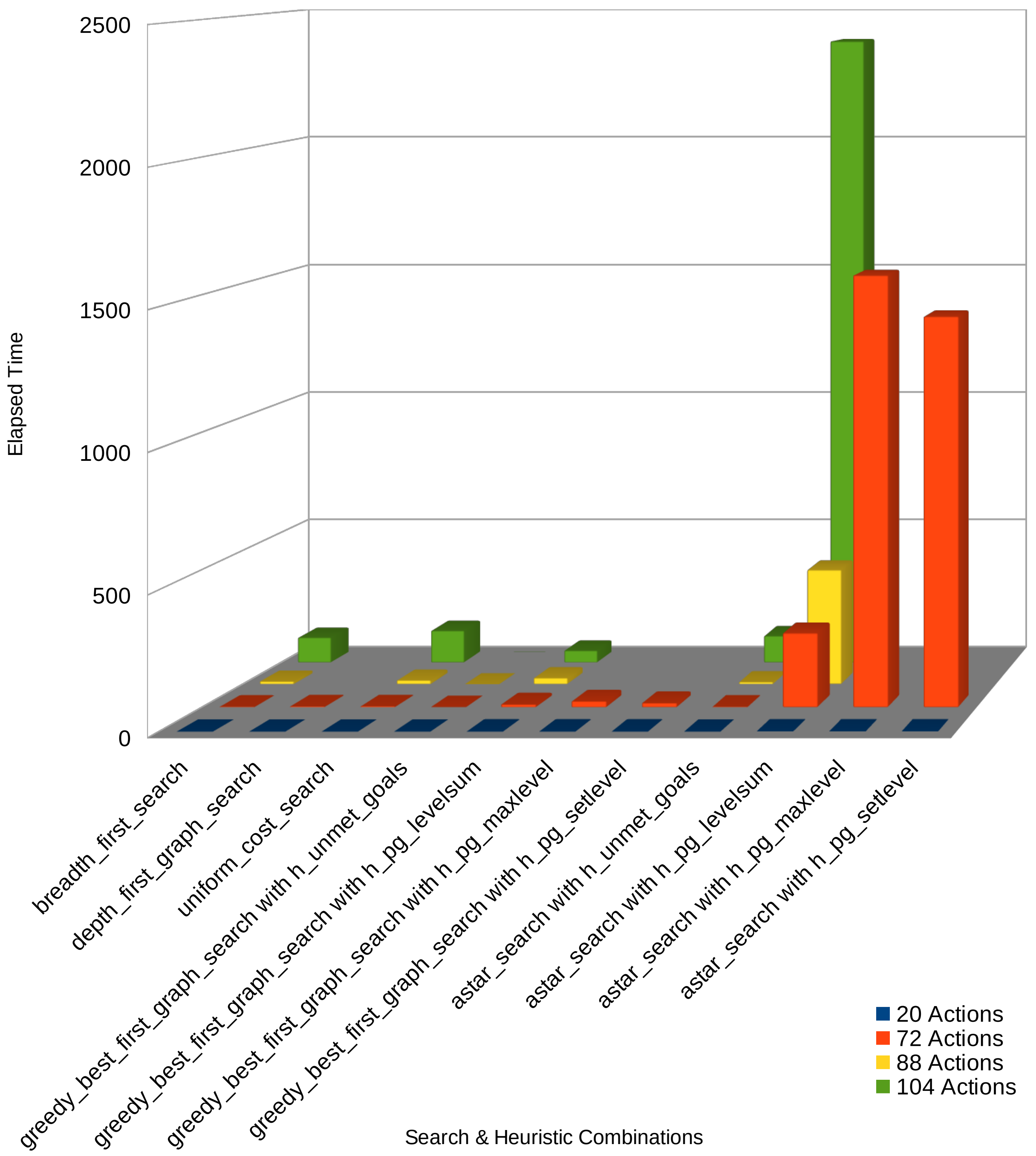
Analysis

Nodes Expanded against the Number of Actions



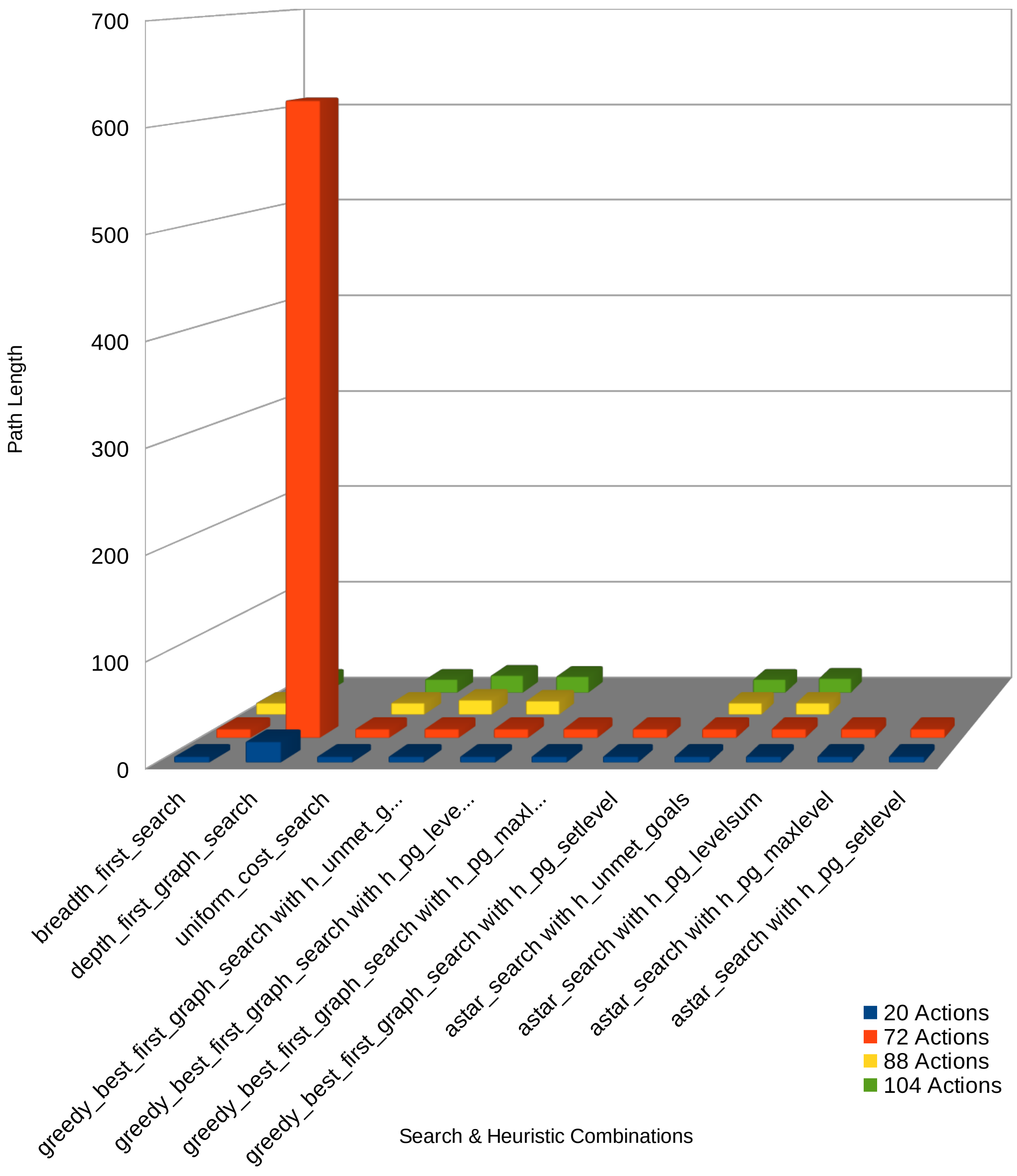
Looking at the previous data and graph, it can be observed that by far the greedy-best-first-search methods presented the less number of expansions, just few of them, actually less that 30, this contrasts with the thousands presented by the uninformed and A* based methods for the problems 3 and 4 (88 and 104 actions).

Search time against the Number of Actions



From the previous data and the graph, it can be seen that in regards the search time, the methods based on greedy-best-first-search performed the best, being the *greedy_best_first_graph_search with h_unmet_goals* the only one that takes less than a second (and it's usable by real time applications) in problem 4, in contrast with the other methods that take several seconds or even minutes (in the case of the ones based on A*).

Path Length against the Number of Actions



From the data from problems 3 and 4 it can, be observed than the length of paths slightly varies across methods, being the uninformed the ones that presented the minor length, followed by the A* ones and lastly by the greedy-best-first-search ones. BFS, uniform-cost-search and A* with *h_unmet_goals* heuristic are the ones that get the minimum path length in problem 4.

Answers to Questions

Q. 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?

From the above results (specially problem 4), the only algorithm which search time was suitable to be used in real time was *greedy_best_first_graph_search* with the *h_unmet_goals* heuristic, which took less than a second in contrast with several seconds or minutes taken by the other methods.

Q. 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)

In general the *greedy_best_first_graph_search* algorithms, seems to perform better for very large domains, since they are able to significantly reduce the amount of expanded nodes, and still being able to produce results close to the optimal.

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

For optimal paths, 3 algorithms were able to find the optimal path: bread-first-search, uniform-cost-search and A* with *h_unmet_goals* heuristic. From which BFS seems to be the quickest one, however the one that expands more nodes.