Project Report: *Build a Forward Planning Agent*

Oliver Koch, 3. May 2020

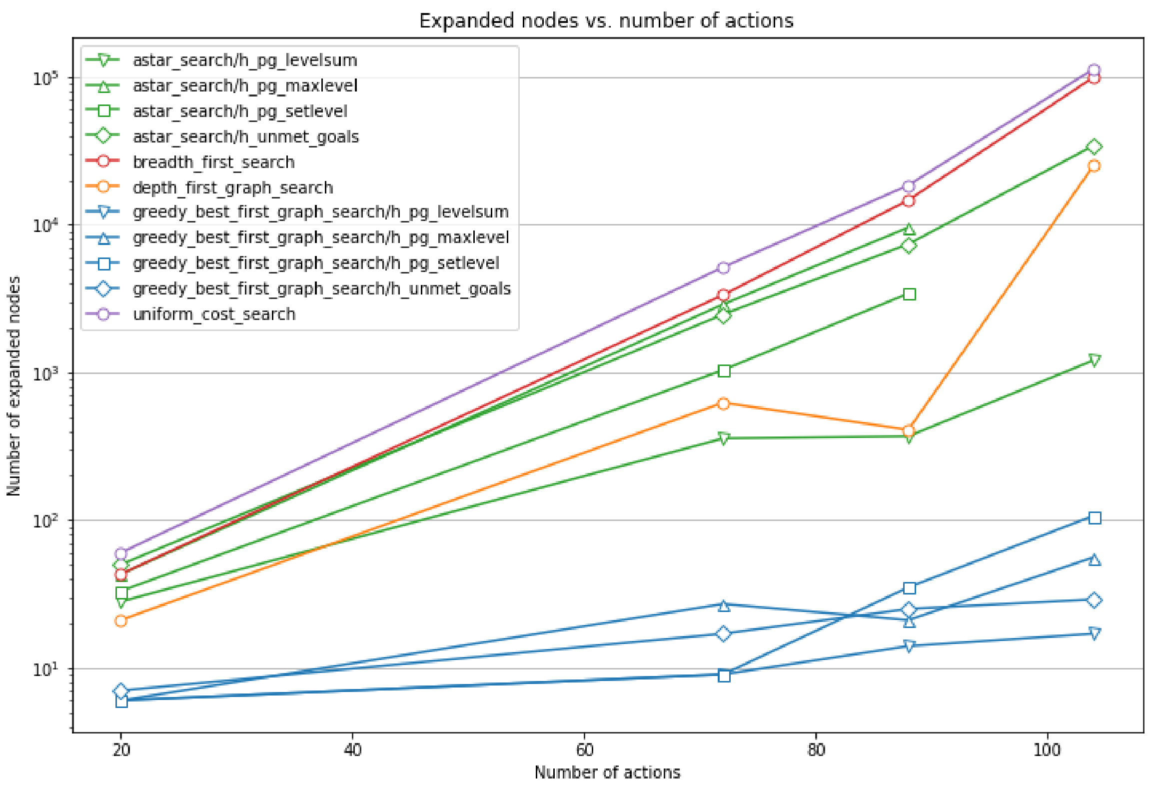
Introduction

All runs were performed locally on a Linux machine using pypy. I attempted to run all algorithms on all problems. I stopped A* searches with set-level and max-level heuristics on air crago problem 4 after approximately 2 hours. All other runs yielded results. The full set of results is shown in the table at the end of the report.

The diagrams below show number ox expansions, search time and length of plan versions number of actions. The number of actions is determined by the problem, see table.

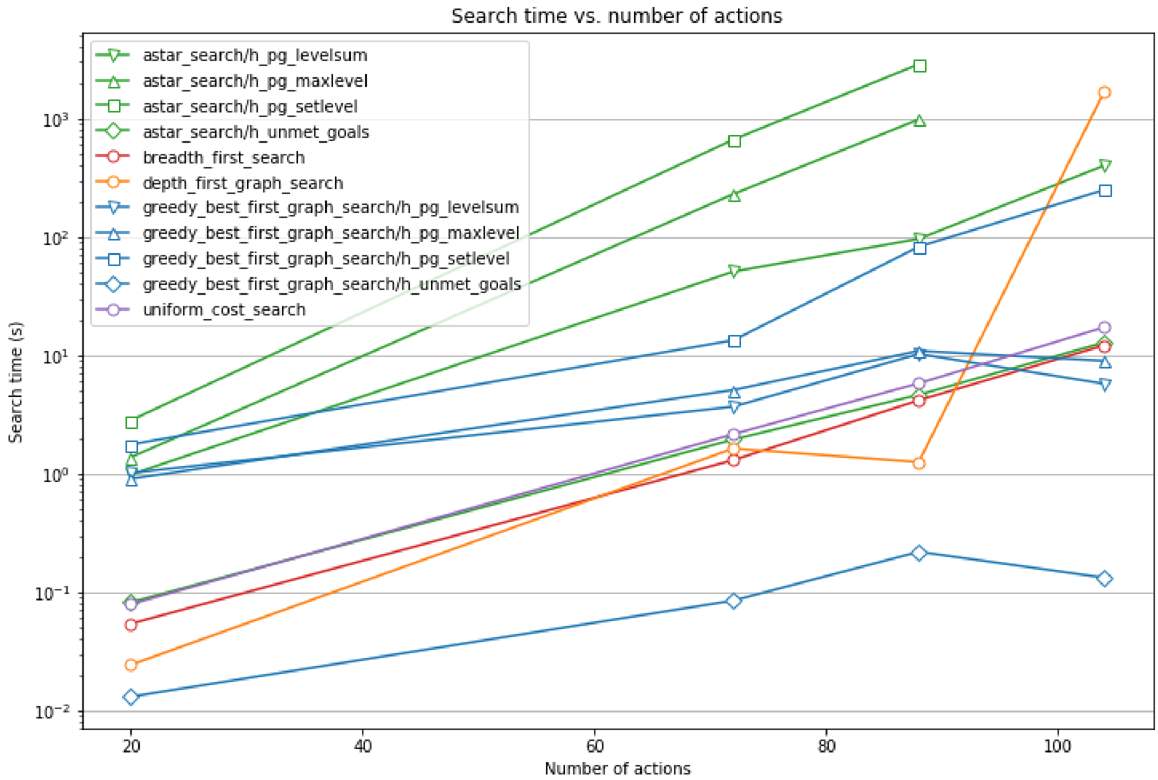
Air Cargo Problem	Number of actions
1	20
2	72
3	88
4	104

Number of expansions



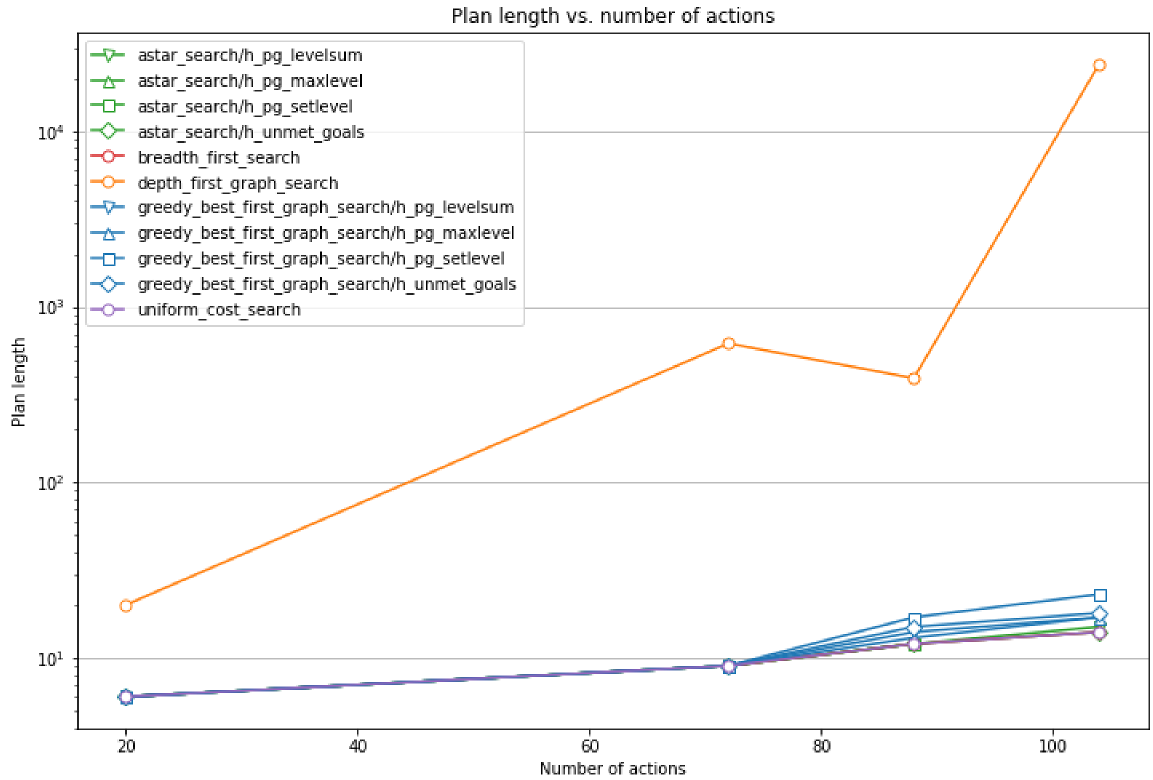
Search time

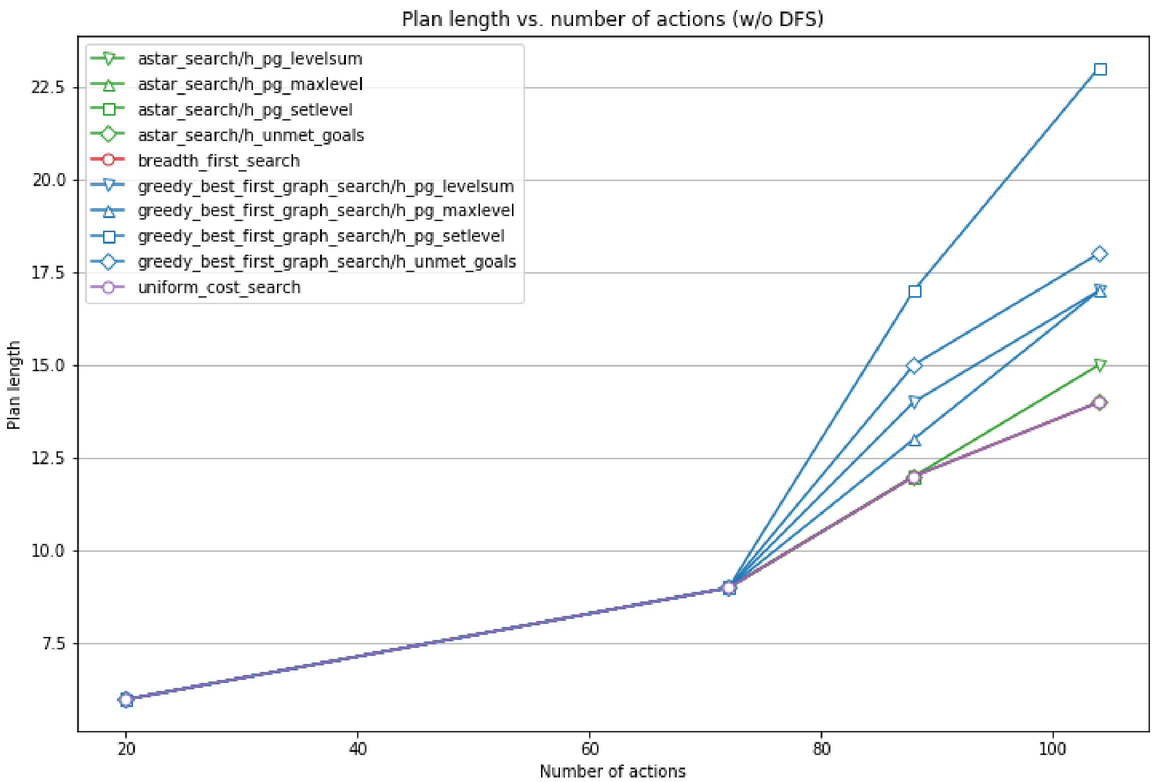
Although A* search variants expands less nodes than BFS/UCS (see figure above), there is no advantage in search time compared to BFS/UCS for heuristics using a planning graph (this may be due to my inefficient implementation of the planning graph).



Plan length

In the first diagram, the very long plans resulting from depth first search dominate and suppress the differences between other algorithms. The second diagram shows the plan length of all algorithms except depth first search.





Question 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?*

Air cargo problem 1 with 20 actions is used as an example for is kind of problems. For this task, space complexity is not relevant. Optimality would be nice but is not required.

Greedy best first graph search with the unmet-gols heuristic has the lowest search time (0.013 s) and would be my first choice for such a situation. The next fastest algorithm is depth-first-search (0.024 s). However, the generated plan is not optimal (20 vs. 6 steps), and I would not use this algorithm. BFS, UCS with might still be acceptable (search time < 0.1 s), if optimality is required and depending on the time available for computation.

Question 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)?*

For this kind of problem, low space complexity is required. BFS, DFS and A* are therefore impractial. **Greedy best-first search** is an approriate algorithm.

According to the AIMA book, also "***RBFS (recursive best-first search)*** *and* ***SMA∗ (simplified memory-bounded A∗)*** *are robust, optimal search algorithms that use limited amounts of memory*".

Question 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** and **A*-Search**(with admissible/consistent heuristics) are optimal. Of the heuristics used here, **max-level** and **set-level** are consistent.

According to the AIMA book, also "***RBFS (recursive best-first search)*** *and* ***SMA∗ (simplified memory-bounded A∗)*** *are robust, optimal search algorithms that use limited amounts of memory*".

Appendix: Results of all program runs

problem	actions	algorithm	heuristic	expansions	goal_tests	new_nodes	plan_length	time
1	20	breadth_first_search		43	56	178	6	0.054103
1	20	depth_first_graph_search		21	22	84	20	0.024324
1	20	uniform_cost_search		60	62	240	6	0.079008
1	20	greedy_best_first_graph_search	h_unmet_goals	7	9	29	6	0.013098
1	20	greedy_best_first_graph_search	h_pg_levelsum	6	8	28	6	1.015349
1	20	greedy_best_first_graph_search	h_pg_maxlevel	6	8	24	6	0.905682
1	20	greedy_best_first_graph_search	h_pg_setlevel	6	8	28	6	1.766840
1	20	astar_search	h_unmet_goals	50	52	206	6	0.082072
1	20	astar_search	h_pg_levelsum	28	30	122	6	0.982791
1	20	astar_search	h_pg_maxlevel	43	45	180	6	1.376201
1	20	astar_search	h_pg_setlevel	33	35	138	6	2.821826
2	72	breadth_first_search		3343	4609	30503	9	1.303654
2	72	depth_first_graph_search		624	625	5602	619	1.632375
2	72	uniform_cost_search		5154	5156	46618	9	2.158004
2	72	greedy_best_first_graph_search	h_unmet_goals	17	19	170	9	0.084858
2	72	greedy_best_first_graph_search	h_pg_levelsum	9	11	86	9	3.683189
2	72	greedy_best_first_graph_search	h_pg_maxlevel	27	29	249	9	5.093418
2	72	greedy_best_first_graph_search	h_pg_setlevel	9	11	84	9	13.405654
2	72	astar_search	h_unmet_goals	2467	2469	22522	9	1.944583
2	72	astar_search	h_pg_levelsum	357	359	3426	9	50.998801
2	72	astar_search	h_pg_maxlevel	2887	2889	26594	9	230.146767
2	72	astar_search	h_pg_setlevel	1037	1039	9605	9	663.803041
3	88	breadth_first_search		14663	18098	129625	12	4.166022
3	88	depth_first_graph_search		408	409	3364	392	1.252732
3	88	uniform_cost_search		18510	18512	161936	12	5.799185
3	88	greedy_best_first_graph_search	h_unmet_goals	25	27	230	15	0.219061

problem	actions	algorithm	heuristic	expansions	goal_tests	new_nodes	plan_length	time
3	88	greedy_best_first_graph_search	h_pg_levelsum	14	16	126	14	10.267711
3	88	greedy_best_first_graph_search	h_pg_maxlevel	21	23	195	13	10.906227
3	88	greedy_best_first_graph_search	h_pg_setlevel	35	37	345	17	82.663379
3	88	astar_search	h_unmet_goals	7388	7390	65711	12	4.637515
3	88	astar_search	h_pg_levelsum	369	371	3403	12	96.208736
3	88	astar_search	h_pg_maxlevel	9580	9582	86312	12	982.185796
3	88	astar_search	h_pg_setlevel	3423	3425	31596	12	2872.003272
4	104	breadth_first_search		99736	114953	944130	14	12.160438
4	104	depth_first_graph_search		25174	25175	228849	24132	1677.166702
4	104	uniform_cost_search		113339	113341	1066413	14	17.259971
4	104	greedy_best_first_graph_search	h_unmet_goals	29	31	280	18	0.132494
4	104	greedy_best_first_graph_search	h_pg_levelsum	17	19	165	17	5.751898
4	104	greedy_best_first_graph_search	h_pg_maxlevel	56	58	580	17	8.994136
4	104	greedy_best_first_graph_search	h_pg_setlevel	107	109	1164	23	249.458142
4	104	astar_search	h_unmet_goals	34330	34332	328509	14	12.858093
4	104	astar_search	h_pg_levelsum	1208	1210	12210	15	400.510749