Udacity AIND-Build a Forward-Planning Agent

Tianwei LAN

Table of problem 1 results:

Search Algorithms	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
breadth_first_search	20	43	56	178	6	0.0061
depth_first_graph_search	20	21	22	84	20	0.0035
uniform_cost_search	20	60	62	240	6	0.0111
greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.0020
greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0.4826
greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0.3505
greedy_best_first_graph_search with h_pg_setlevel	20	6	8	28	6	0.6377
astar_search with h_unmet_goals	20	50	52	206	6	0.0095
astar_search with h_pg_levelsum	20	28	30	122	6	1.1843
astar_search with h_pg_maxlevel	20	43	45	180	6	1.2165
astar_search with h_pg_setlevel	20	33	35	138	6	1.5182

Table of problem 2 results:

Search Algorithms	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
breadth_first_search	72	3343	4609	30503	9	1.9907
depth_first_graph_search	72	624	625	5602	619	3.1751
uniform_cost_search	72	5154	5156	46618	9	3.3335
greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.0196
greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	10.478
greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	21.183
greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	9	15.692
astar_search with h_unmet_goals	72	2467	2469	22522	9	2.2725
astar_search with h_pg_levelsum	72	357	359	3426	9	264.75
astar_search with h_pg_maxlevel	72	2887	2889	26594	9	1530.4
astar_search with h_pg_setlevel	72	1037	1039	9605	9	1410.7

From the first two problems, I find that depth_first_graph_search has longer plan length than other search algorithms, which makes its solution not optimal. Besides, uniform_cost_search has large node expansions, which uses more memory. In terms of greedy_best_first_graph_search, greedy_best_first_graph_search with h_pg_maxlevel seems to have larger node expansions and longer search time. Speaking of astar_search, astar_search with h_pg_maxlevel and astar_search with h_pg_setlevel experience much longer search time than other astar_search algorithms. Therefore, I exclude the depth_first_graph_search, uniform_cost_search, greedy_best_first_graph_search with h_pg_maxlevel, astar_search with h_pg_maxlevel and astar_search with h_pg_setlevel for solving problem 3 and 4.

Table of problem 3 results:

Search Algorithms	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
breadth_first_search	88	14663	18098	129625	12	10.679
greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0.0361
greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	23.496
greedy_best_first_graph_search with h_pg_setlevel	88	35	37	345	17	87.477
astar_search with h_unmet_goals	88	7388	7390	65711	12	8.5155
astar_search with h_pg_levelsum	88	369	371	3403	12	427.79

Table of problem 4 results:

Search Algorithms	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time (s)
breadth_first_search	104	99736	114953	944130	14	97.664
greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	18	0.0596
greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	42.529
greedy_best_first_graph_search with h_pg_setlevel	104	107	109	1164	23	396.01
astar_search with h_unmet_goals	104	34330	34332	328509	14	56.446
astar_search with h_pg_levelsum	104	1208	1210	12210	15	2402.1

From problem 3 and 4, I find that breadth_first_search and astar_search with h_unmet_goals have optimal solutions with shortest plan length, but they have very large node expansions. Astar_search with h_pg_levelsum has almost optimal solution, but suffers from a very long search time. Greedy_best_first_graph_search with h_unmet_goals has low node expansions as well as a very little search time, but its plan length is quite long. Greedy_best_first_graph_search with h_pg_levelsum uses the least node expansions, but it does not find the shortest plan length. Greedy_best_first_graph_search with h_pg_setlevel has the longest plan length for problem 3 and 4, so it is not a good algorithm to find the optimal solution.

In terms of **nodes expansions**, greedy_best_first_graph_search algorithms always have less nodes expansions and take up less memory compared to other algorithms even if the problem size increases. On the contrary, the nodes expansions of breadth_first_search and astar_search with h_unmet_goals increase dramatically as the problem size increases.

With regards to **search time**, uninformed searches have similar search speed for the same problem and the search time rises gradually when the problem size increases. When the actions increase, the differences among greedy_best_first_graph_search become more and more evident, especially greedy_best_first_graph_search with h_unmet_goals always has so little search time compared to other algorithms. Finally, the search time of astar_search algorithms increases sharply when the problem size increases except for astar_search with h_unmet_goals.

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?

The requirements of this question are a small number of actions and a very short search time. Based on these constraints, breadth_first_search and greedy_best_first_graph_search with h_unmet_goals seem to be good algorithms.

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)

The requirements of this question are a very big problem size and an optimal solution for planning. Based on these constraints, astar_search with h_pg_levelsum seems to be a good algorithm because of its optimal solution for planning and fewer node expansions.

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

The requirement of this question is to find only optimal plans. Based on this constraint, breadth_first_search seems to be a good algorithm.