Project 2: Build a Forward Planning Agent

Contents

Table of Figures

Search complexity as a function of domain size

The diagram (**Figure 1: Expansions by # Actions**) shows the number of nodes expanded during search for each of the search-heuristic combinations across the four problem domains.

Having tested with Problems 1 and 2 it became very clear that some search – heuristic combinations grew significantly in complexity as the # Actions was expanded from 20 to 72. To progress, I chose those searches that appeared to exhibit the lowest growth in search complexity to test against Problems 3 and 4. Specifically, I chose the following search – heuristic combinations to evaluate against Problems 3 and 4;

- depth_first_graph_search
- greedy_best_first_graph_search with h_pg_levelsum
- greedy_best_first_graph_search with h_pg_setlevel
- astar_search with h_pg_levelsum
- astar_search with h_pg_setlevel

These constitute one **Uninformed Search**, two **Greedy Best First Searches** and two **A* Searches**. Of these, the following searches seems to exhibit the least growth in search complexity across the four problems:

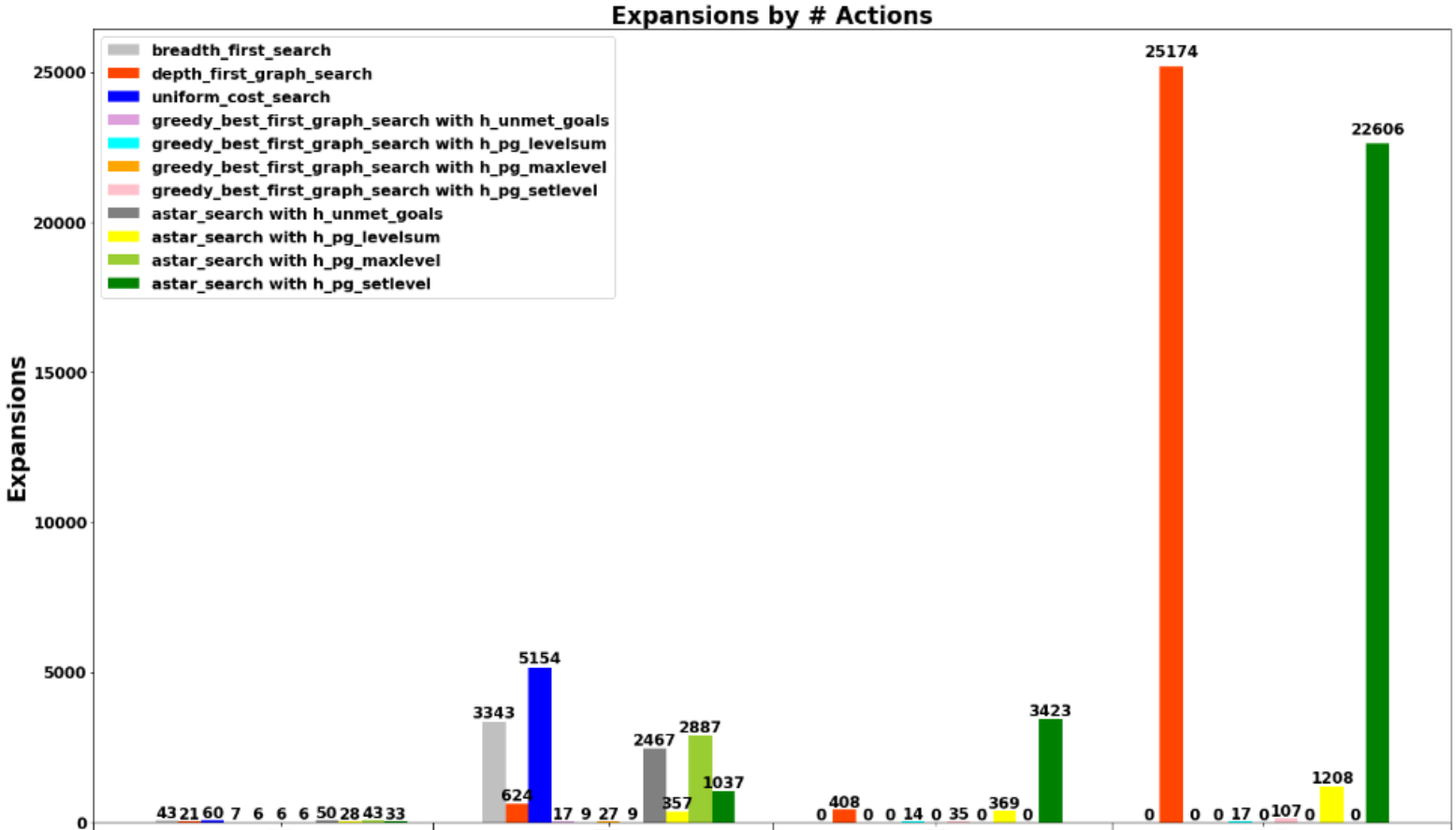
- greedy_best_first_graph_search with h_pg_levelsum (17 expanded nodes for problem 4)
- greedy_best_first_graph_search with h_pg_setlevel (107 expanded nodes for problem 4)
- astar_search with h_pg_levelsum (1208 expanded nodes for problem 4)

Greedy Best First Graph Search with either heuristic significantly reduced search complexity when compared to other search algorithms. The version using the simpler **level sum heuristic** expands fewer nodes than that using **set level**.

The **A* Search with level sum heuristic** performed well in terms of search complexity but as can be seen from **Figure 4: Path Length by # Actions**, this search did not find the optimal path to the goal for problem 4, presumably since it is using an inadmissible heuristic.

A* Search with set level heuristic performed less well in terms of search complexity (22606 expanded nodes for Problem 4) but as can be seen from **Figure 4: Path Length by # Actions** it did indeed find an optimal path to the goal for all problems.

Deep First Search performed poorly in terms of search complexity and as can be seen from **Figure 4: Path Length by # Actions** fails abysmally for problem 4 to find an optimal path to the goal.



	20	72	88	104
breadth_first_s	43	3343		
depth_first_graph_s	21	624	408	25174
uniform_cost_s	60	5154		
greedy_best_first_graph_s w h_unmet_goals	7	17		
greedy_best_first_graph_s w h_pg_levelsum	6	9	14	17
greedy_best_first_graph_s w h_pg_maxlevel	6	27		
greedy_best_first_graph_s w h_pg_setlevel	6	9	35	107
astar_s w h_unmet_goals	50	2467		
astar_s w h_pg_levelsum	28	357	369	1208
astar_s w h_pg_maxlevel	43	2887		
astar_s w h_pg_setlevel	33	1037	3423	22606

Figure 1: Expansions by # Actions

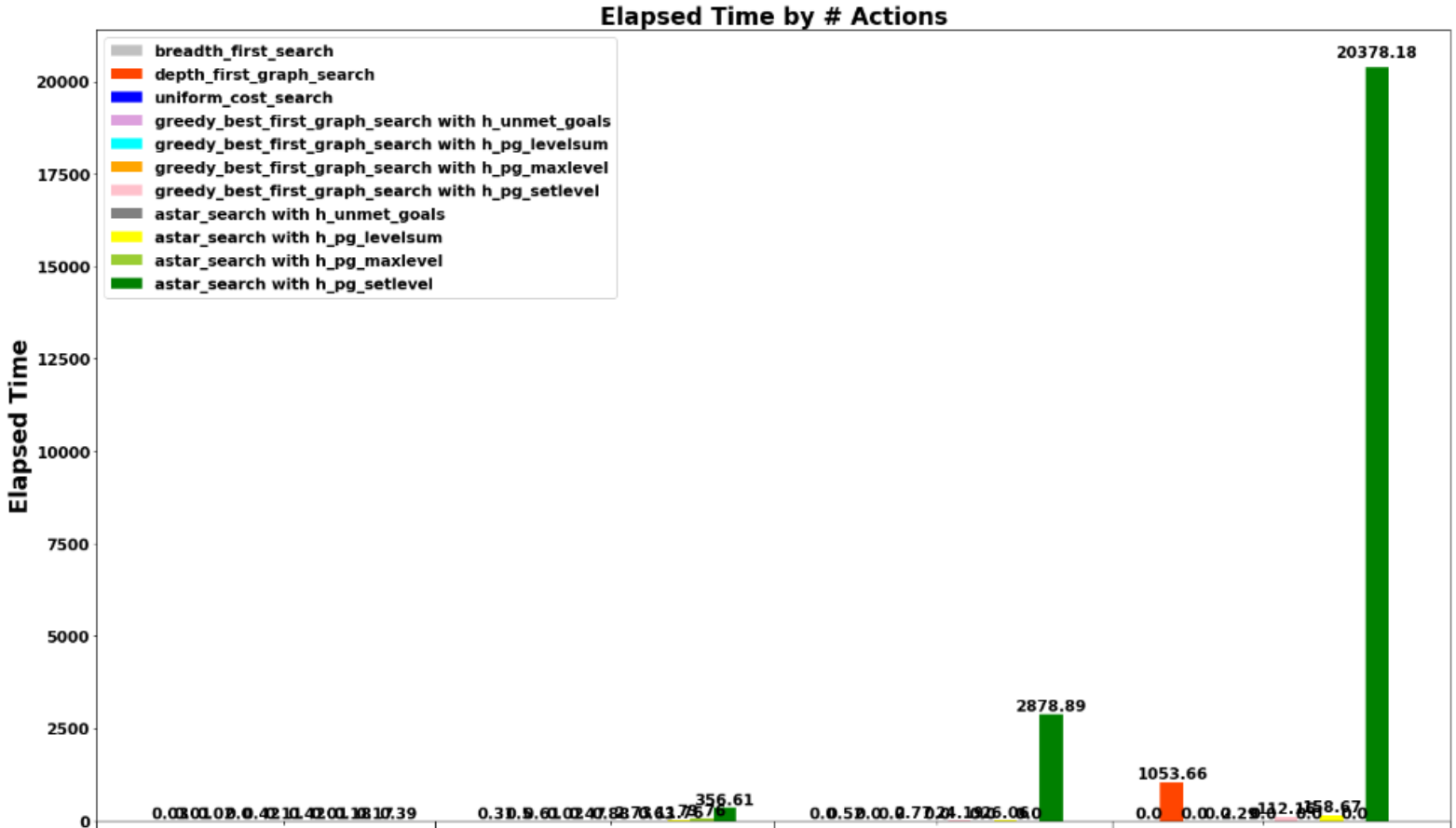
Search time against the number of actions in the domain

The Search execution time varied very significantly indeed across the different searches as the problem space increased, as can be seen in **Figure 2: Elapsed Time (Seconds) by # Actions**. In fact, the variation in search times was so great that it was necessary to use a logarithmic graph of search times as per **Figure 3: Natural Log of Elapsed Time by # Actions** to gain a better appreciation of the relative performance of the various searches and heuristics.

A* Search with Set Level heuristic performed relatively poorly as the search space increased, however as can be seen in **Figure 4: Path Length by # Actions**, it was the only search that found the optimal path to the goal for all 4 problems.

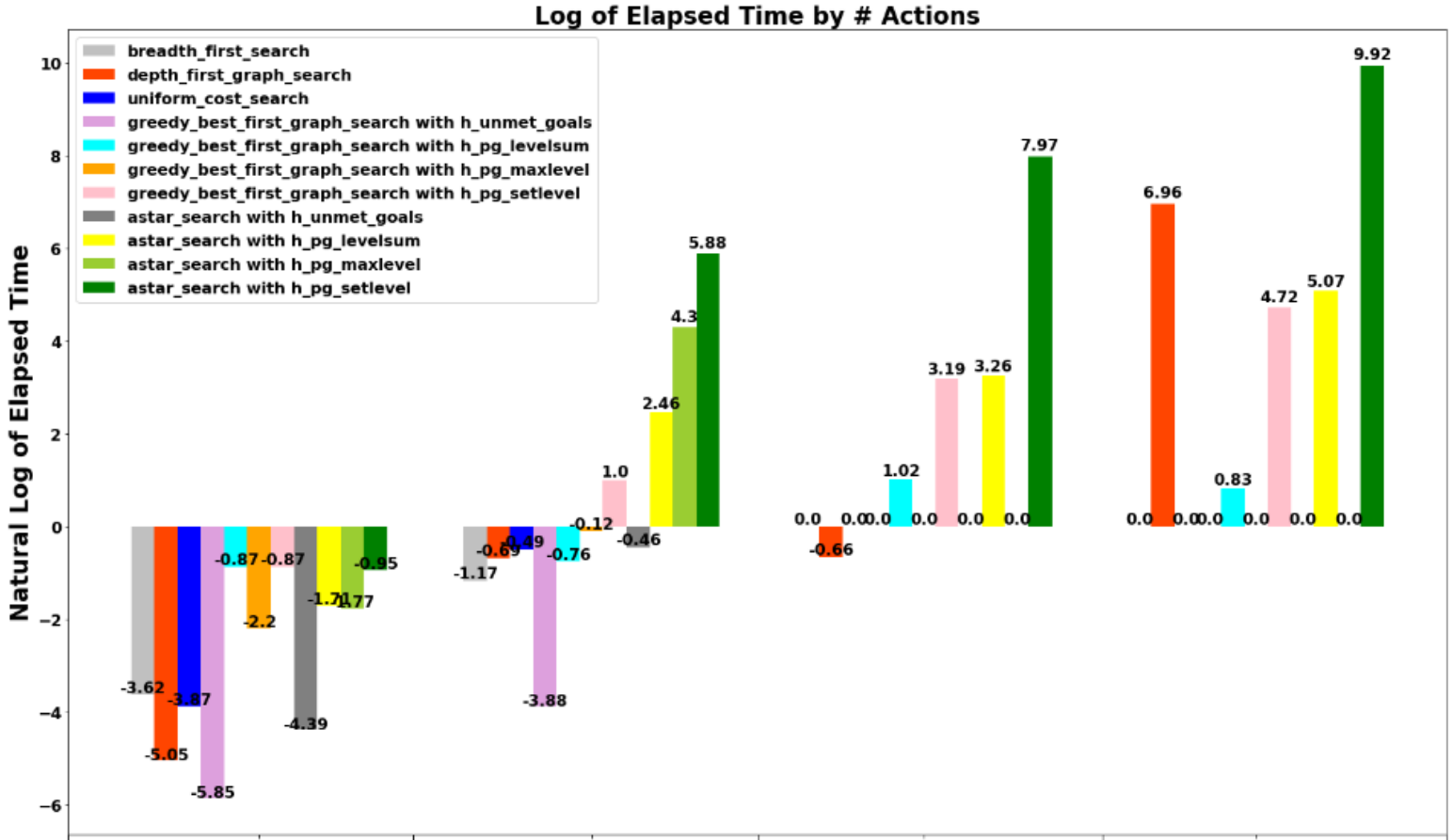
As would expected, **Depth First Search** performance degraded significantly as the search space grew.

Other searches such as **A* Search with Level Sum Heuristic**, **Greedy Best First Search with Level Sum** and **Greedy Best First Search with Set Level** all performed very well as the search space increased and as can be seen from **Figure 4: Path Length by # Actions** most found paths that were close to optimal across all 4 problems.



	20	72	88	104
breadth_first_s	0.03	0.31		
depth_first_graph_s	0.01	0.5	0.52	1053.66
uniform_cost_s	0.02	0.61		
greedy_best_first_graph_s w h_unmet_goals	0.0	0.02		
greedy_best_first_graph_s w h_pg_levelsum	0.42	0.47	2.77	2.29
greedy_best_first_graph_s w h_pg_maxlevel	0.11	0.88		
greedy_best_first_graph_s w h_pg_setlevel	0.42	2.73	24.19	112.16
astar_s w h_unmet_goals	0.01	0.63		
astar_s w h_pg_levelsum	0.18	11.76	26.06	158.67
astar_s w h_pg_maxlevel	0.17	73.76		
astar_s w h_pg_setlevel	0.39	356.61	2878.89	20378.18

Figure 2: Elapsed Time (Seconds) by # Actions



	20	72	88	104
breadth_first_s	0.03	0.31		
depth_first_graph_s	0.01	0.5	0.52	1053.66
uniform_cost_s	0.02	0.61		
greedy_best_first_graph_s w h_unmet_goals	0.0	0.02		
greedy_best_first_graph_s w h_pg_levelsum	0.42	0.47	2.77	2.29
greedy_best_first_graph_s w h_pg_maxlevel	0.11	0.88		
greedy_best_first_graph_s w h_pg_setlevel	0.42	2.73	24.19	112.16
astar_s w h_unmet_goals	0.01	0.63		
astar_s w h_pg_levelsum	0.18	11.76	26.06	158.67
astar_s w h_pg_maxlevel	0.17	73.76		
astar_s w h_pg_setlevel	0.39	356.61	2878.89	20378.18

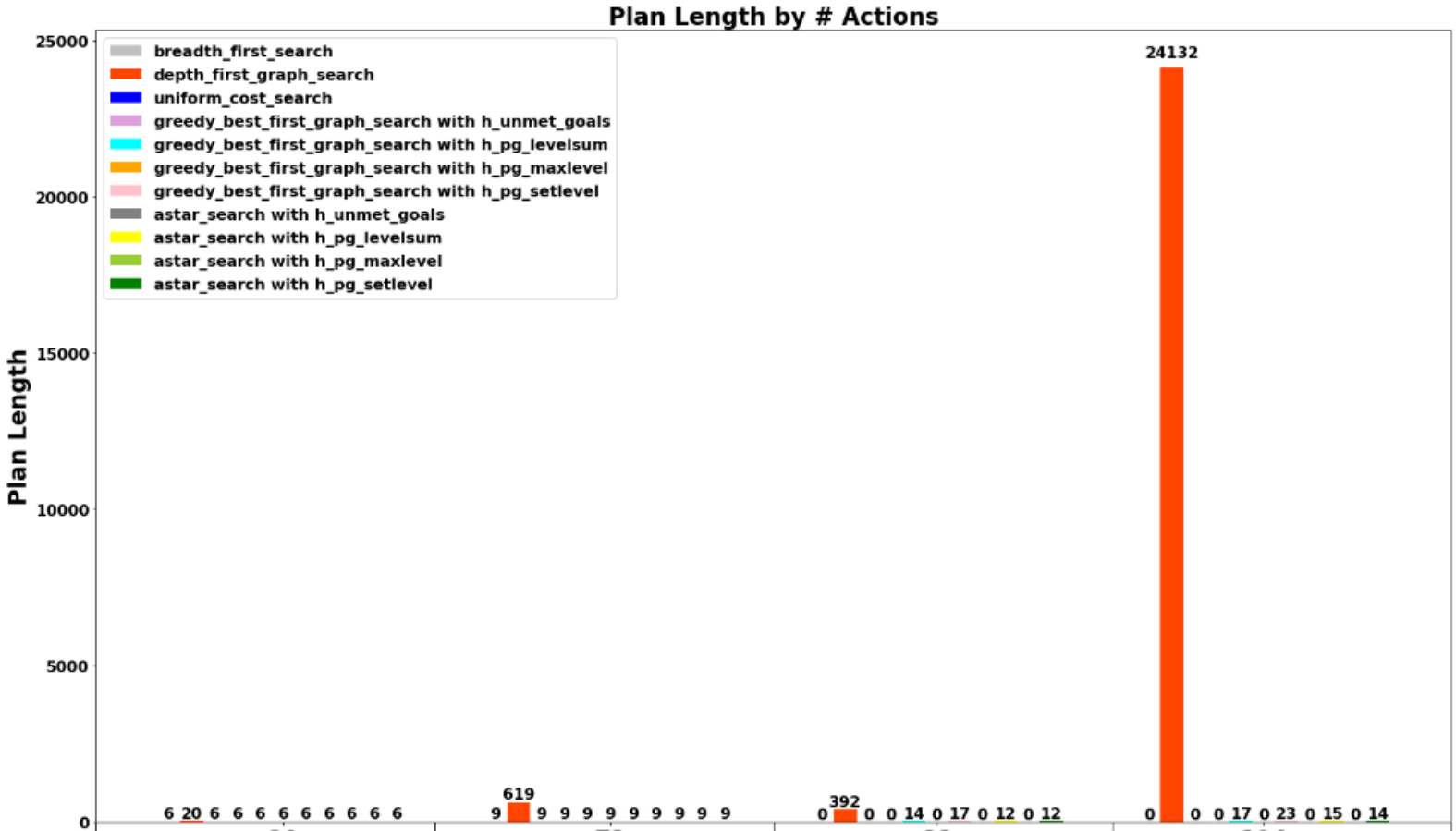
Figure 3: Natural Log of Elapsed Time by # Actions

Optimality of solution as a function of domain size, search algorithm and heuristic.

Figure 4: Path Length by # Actions shows the length of path found by each search – heuristic combination across each of the search problems.

As can be seen from this chart, of the searches executed for all problems, **A* Search with Level Set heuristic** was the only search – heuristic combination to find an optimal path for all four problems. This is presumably because that search is **complete** and the heuristic is **admissible**.

Interestingly **A* Search with Level Sum heuristic** performed very well in terms of finding near-optimal paths to the goals, as did **Greedy Best First Search with Level Sum**. Both searches performed much better than the optimal **A* Search** in terms of execution time and the # of expanded nodes so the advantages of these approaches for this problem domain seem significant, albeit at the cost of less than non-optimal path length. With that in mind **A* Search with Level Sum Heuristic** seems like a very good choice for this problem domain given that it performs very well and delivers optimal or near optimal paths to the goals in all four of the problems.



	20	72	88	104
breadth_first_s	6	9		
depth_first_graph_s	20	619	392	24132
uniform_cost_s	6	9		
greedy_best_first_graph_s w h_unmet_goals	6	9		
greedy_best_first_graph_s w h_pg_levelsum	6	9	14	17
greedy_best_first_graph_s w h_pg_maxlevel	6	9		
greedy_best_first_graph_s w h_pg_setlevel	6	9	17	23
astar_s w h_unmet_goals	6	9		
astar_s w h_pg_levelsum	6	9	12	15
astar_s w h_pg_maxlevel	6	9		
astar_s w h_pg_setlevel	6	9	12	14

Figure 4: Path Length by # Actions

Questions Posed

Question 1: Restricted Domain

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: I would choose the **Greedy Best First Search with Level Sum heuristic** for such a problem domain. It seems to perform much better than A* Search yet still manages to deliver near optimal paths to the goal.

Question 2: Large Domain

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 a very large domain we would want the most optimal search paths possible but would need to be aware of the fact that there would be a limited time each day to compute the plans. In such a scenario **A* Search with Level Sum** heuristic would seem like the best choice. It delivers near optimum paths with very good search performance.

Question 4: Optimal Plans

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

Answer: To find optimal plans one should use a **complete** search such as the **A* Search** in conjunction with **Admissible Heuristics** (such as level set). As can be seen from the results above, when **A* Search** is used with inadmissible heuristics, such as level sum, it can deliver non-optimal paths.