ARTIFICIAL INTELLIGENCE NANODEGREE

Classical Planning Project Report

Author: Vraj Parikh

Problem 1

Number of actions: 20

Search Comparison

Search Strategy	Heuristic	Number of Nodes Expanded	Is Optimal?	Path Length	Execution Time
breadth_first_search		43	Y	6	0.00698
depth_first_graph_search		21	N	20	0.00363
uniform_cost_search		60	Y	6	0.00966
greedy_best_first_graph_search	h_unmet_goals	7	Y	6	0.00144
	h_pg_levelsum	6	Y	6	0.15597
	h_pg_maxlevel	6	Y	6	0.10780
	h_pg_setlevel	6	Y	6	0.52557
astar_search	h_unmet_goals	50	Y	6	0.00817
	h_pg_levelsum	28	Y	6	0.37691
	h_pg_maxlevel	43	Y	6	0.37708
	h_pg_setlevel	33	Y	6	1.18008

Example Optimal Path (greedy_best_first_graph_search with h_pg_maxlevel)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Load(C1, P2, SFO)

Fly(P2, SFO, JFK)

Unload(C1, P2, JFK)

Problem 2

Number of actions: 72

Search Comparison - Tabular

Search Strategy	Heuristic	Number of Nodes Expanded	Is Optimal?	Path Length	Execution Time
breadth_first_search		3343	Y	9	4.14653
depth_first_graph_search		624	N	619	6.42167
uniform_cost_search		5154	Y	9	6.66452
greedy_best_first_graph_search	h_unmet_goals	17	Y	9	0.04178
	h_pg_levelsum	9	Y	9	7.87911
	h_pg_maxlevel	27	Y	9	10.43481
	h_pg_setlevel	9	Y	9	16.35626
astar_search	h_unmet_goals	2467	Y	9	2.26385
	h_pg_levelsum	357	Y	9	106.11423
	h_pg_maxlevel	2887	Y	9	617.346
	h_pg_setlevel	1037	Y	9	1478.675

Example Optimal Path (greedy_best_first_graph_search with h_pg_maxlevel)

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Load(C3, P3, ATL)

Fly(P2, JFK, SFO)

Fly(P3, ATL, SFO)

Fly(P1, SFO, JFK)

Unload(C3, P3, SFO)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Problem 3

Number of actions: 88

Search Comparison - Tabular

Search Strategy	Heuristic	Number of Nodes Expanded	Is Optimal?	Path Length	Execution Time
breadth_first_search		14663	Y	12	11.0187
greedy_best_first_graph_search	h_unmet_goals	25	N	15	0.03951
	h_pg_levelsum	14	N	14	8.69650
astar_search	h_unmet_goals	7388	Y	12	9.29428
	h_pg_levelsum	369	Y	12	175.677

Example Optimal Path (astar_search with h_pg_levelsum)

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Problem 4

Number of actions: 104

Search Comparison - Tabular

Search Strategy	Heuristic	Number of Nodes Expanded	Is Optimal?	Path Length	Execution Time
breadth_first_search		99736	Y	14	88.9396
greedy_best_first_graph_search	h_unmet_goals	29	Y	18	0.05500
	h_pg_levelsum	17	Y	17	13.84768
astar_search	h_unmet_goals	34330	Y	14	50.44404
	h_pg_levelsum	1208	Y	14	935.2848

Example Optimal Path (astar_search with h_pg_levelsum)

Load(C1, P1, SFO)

Fly(P1, SFO, ORD)

Load(C4, P1, ORD)

Load(C5, P1, ORD)

Fly(P1, ORD, JFK)

Unload(C5, P1, JFK)

Unload(C1, P1, JFK)

Load(C2, P1, JFK)

Fly(P1, JFK, SFO)

Fly(P2, JFK, ATL)

Load(C3, P2, ATL)

Fly(P2, ATL, JFK)

Unload(C4, P1, SFO)

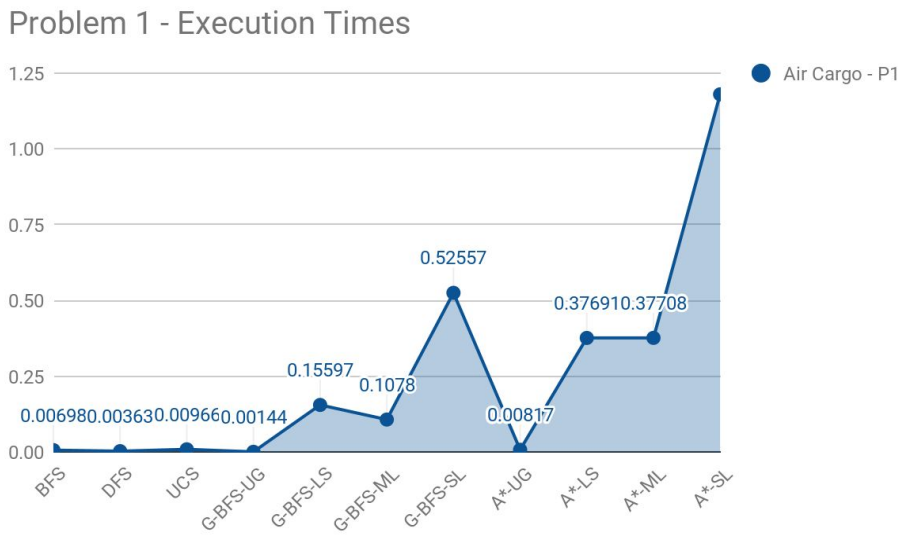
Unload(C3, P2, JFK)

Unload(C2, P1, SFO)

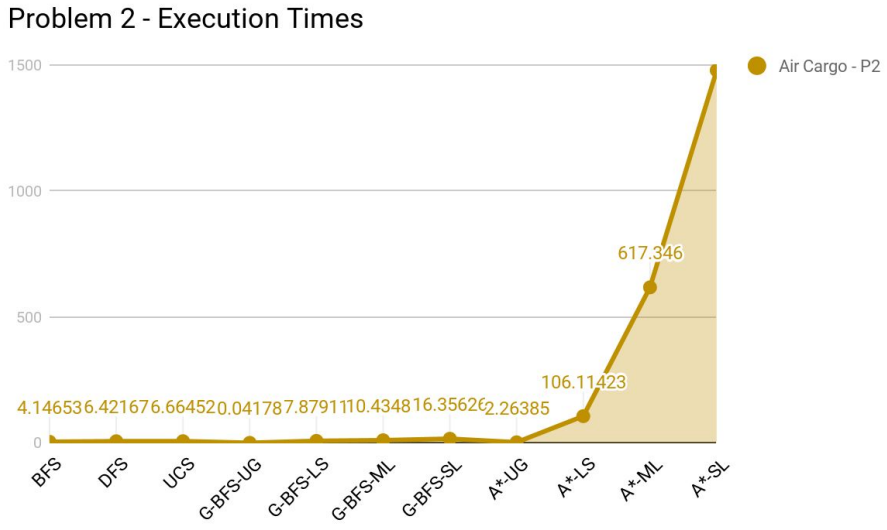
Charts

Execution Time

Problem 1

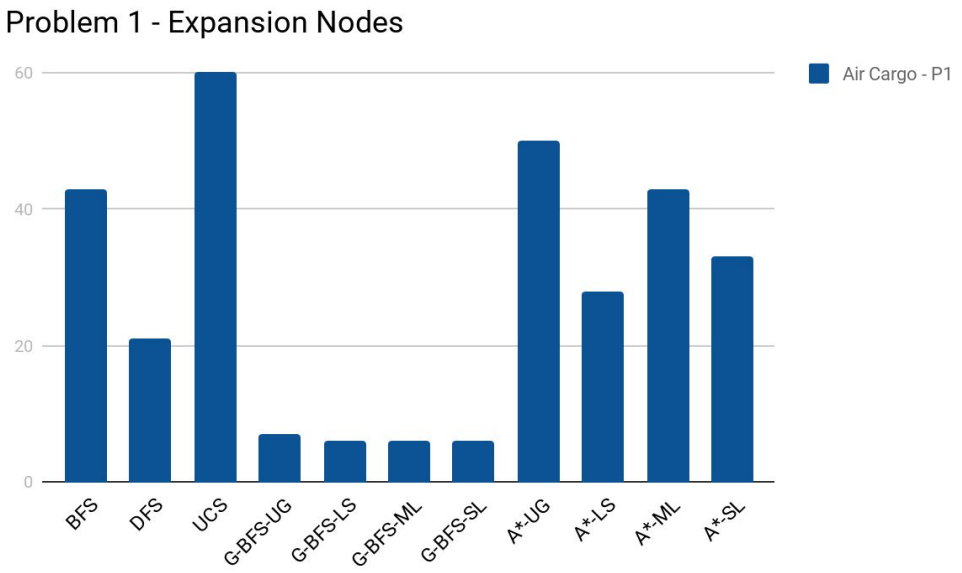


Problem 2

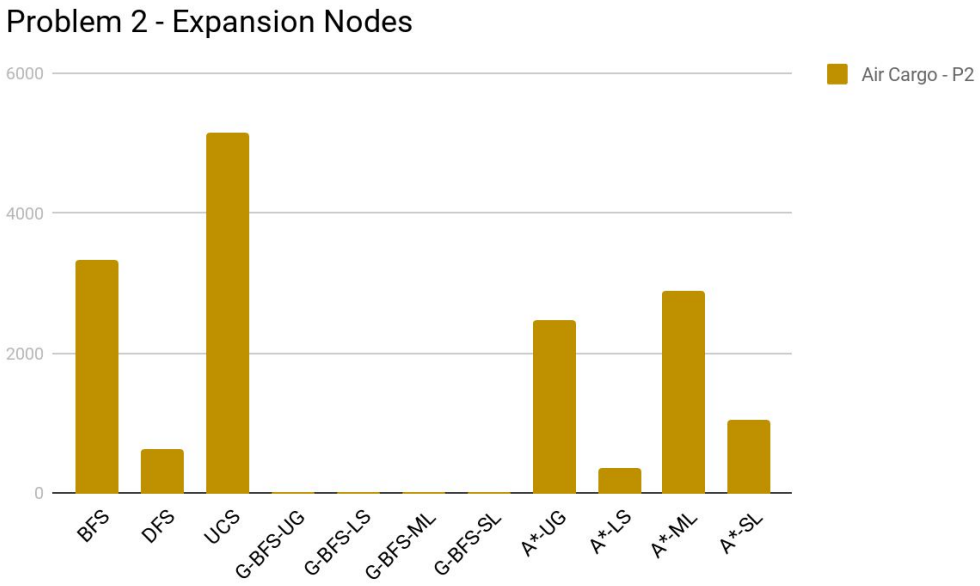


Expansion Nodes

Problem 1



Problem 2

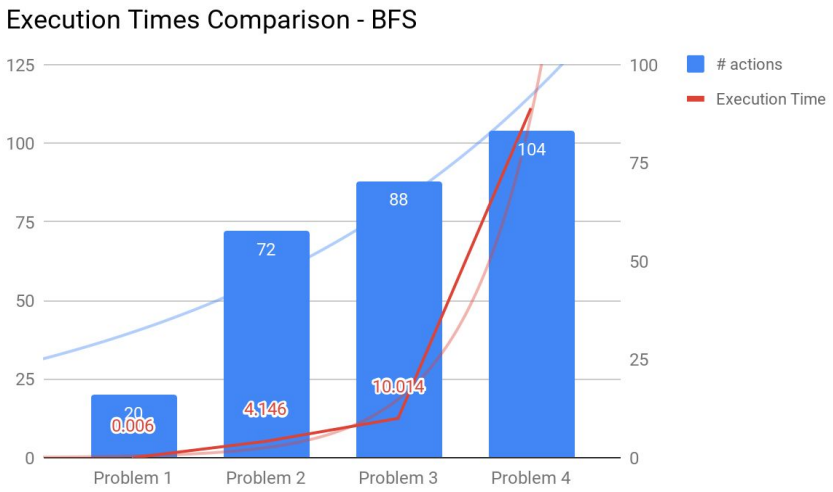


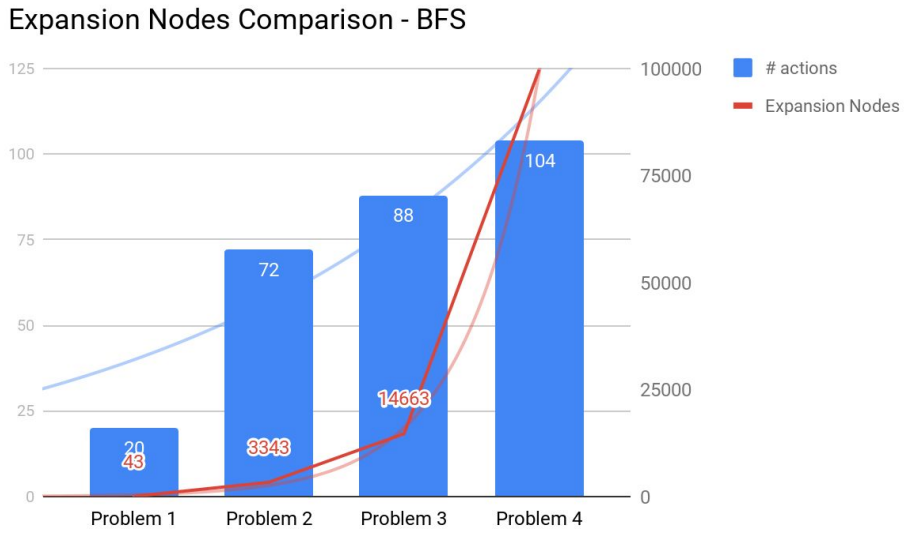
Problems Comparison

BFS

The execution times as well as number of expansion nodes increases exponentially as the size of the problem size increases.

Charts below show trends for execution times and expansion nodes, as a function of number of action nodes,

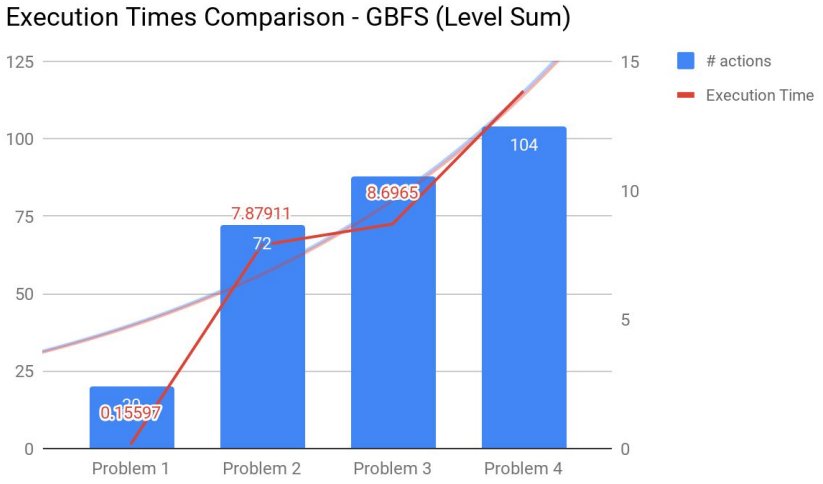


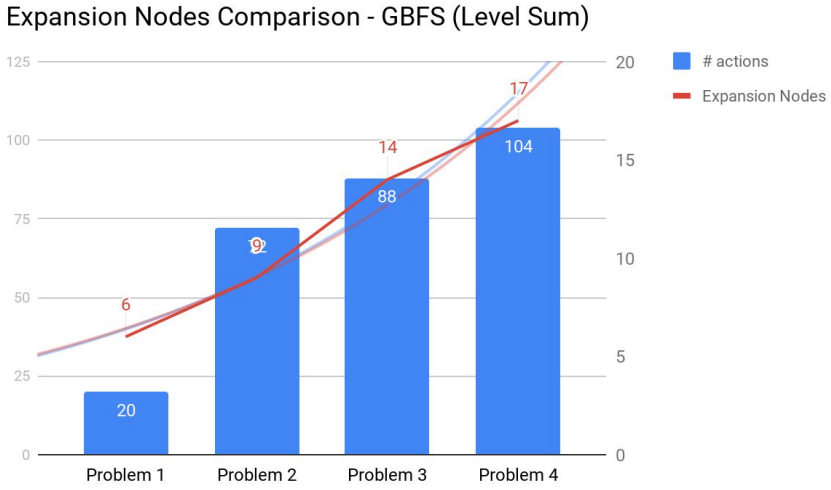


Greedy Best First Search - Level Sum Heuristic

The execution times as well as number of expansion nodes increases linearly as the size of the problem size increases. So the Greedy BFS seems to constantly work, even though the problem size increases. The only problem here is that as the problem size increases it does not give the most optimal solution.

Charts below show trends for execution times and expansion nodes, as a function of number of action nodes.

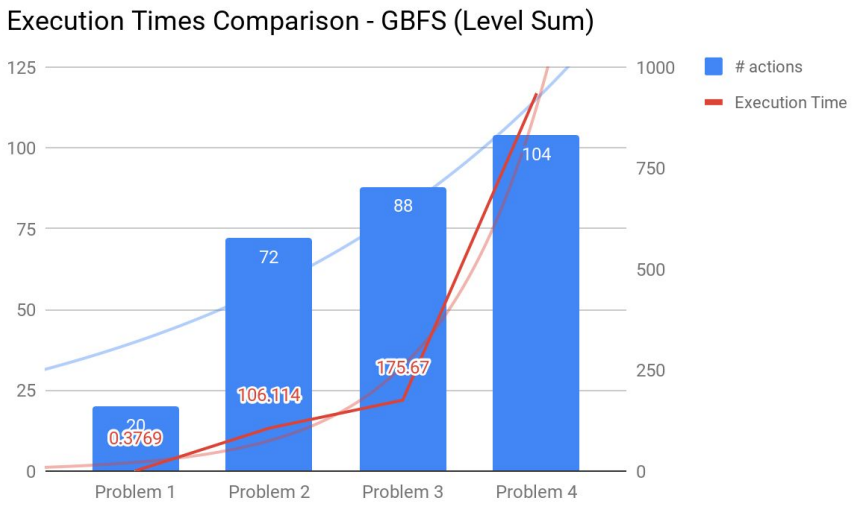


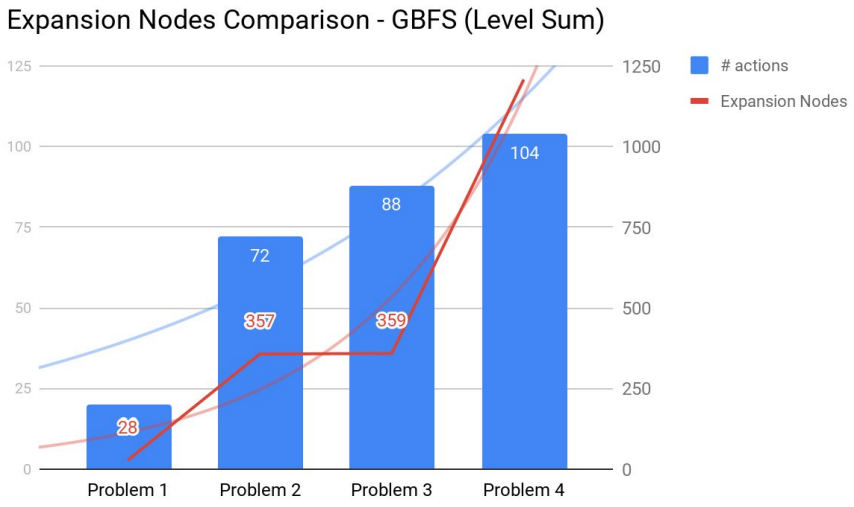


A* Search - Level Sum Heuristic

The graph here looks much steeper than Greedy BFS, but less steeper than the BFS or any of the uninformed searches. Also, unlike Greedy BFS, A* guranties to give most optimal solution.

Charts below show trends for execution times and expansion nodes, as a function of number of action nodes.





Questions

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?**

   For a very restricted domain, it seems that uninformed search works really well, especially - DFS in terms of both execution time and number of nodes that are expanded. DFS won't give the least path, but it is guaranteed to give answers with least expansions. If least path is needed, then BFS or UCS would be suitable.

   For A* & Greedy BFS, a good heuristic function would be needed, that might change from problems to problems.

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)**

   If least path is not a concern, then Greedy BFS with some heuristic works best for very large domains. It is best in terms of both time and space. If optimality is a concern, then A* with some heuristic would work best here.

   For both scenarios, heuristic can be selected based on the problem.

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

   For this scenario, A* seems to be the best fit. Also, level sum / unmet goals heuristic seems to work very well in terms of execution time for the Air Cargo problem.