Heuristic Analysis

Planning Search

Results

Summary Table

Search Strategy Name	Problem 1 Duration	1 Expansions	1 Goal Tests	1 New Nodes	1 Optimal Plan Length	Problem 2 Duration	2 Expansions	2 Goal Tests	2 New Nodes	2 Optimal Plan Length
breadth_first_tree_search	2.01	1,458	1,459	5,960	6					
recursive_best_first_search with h_1	6.77	4,429	4,230	17,023	6					
astar_search with h_1	0.09	55	57	224	6	134.18	4,853	4,855	44,041	9
astar_search with h_ignore_preconditions	0.13	41	43	170	6	43.81	1,506	1,508	13,820	9
astar_search with h_pg_levelsum	5.85	11	13	50	6	825.51	86	88	841	9
breadth_first_search	0.08	43	56	180	6	50.44	3,343	4,609	30,509	9
uniform_cost_search	0.17	55	57	224	6	123.76	4,853	4,855	44,041	9
greedy_best_first_graph_search with h_1	0.06	7	9	28	6	19.39	998	1,000	8,982	21
depth_limited_search	0.19	101	271	414	50					
depth_first_graph_search	0.02	21	22	84	20	8.26	624	625	5,602	875
Problem Performance	1.54	622	622	2,436	12	172.19	2,323	2,506	21,119	134

Search Strategy Name	Problem 3 Duration	3 Expansions	3 Goal Tests	3 New Nodes	3 Optimal Plan Length	Search Performance Duration	Search Performance Expansions	Search Performance Goal Tests	Search Performance New Nodes	Search Performance Optimal Plan Length
breadth_first_tree_search						2.01	1,458	1,459	5,960	6
recursive_best_first_search with h_1						6.77	4,429	4,230	17,023	6
astar_search with h_1	506.59	11,482	11,484	85,785	12	213.62	5,463	5,465	43,350	9
astar_search with h_ignore_preconditions	72.08	2,494	2,496	19,532	12	38.67	1,347	1,349	11,174	9
astar_search with h_pg_levelsum	2,102.74	306	308	2,148	12	978.03	134	136	1,013	9
breadth_first_search	137.47	8,062	11,196	64,308	12	62.66	3,816	5,287	31,666	9
uniform_cost_search	414.75	11,482	11,484	85,785	12	179.56	5,463	5,465	43,350	9
greedy_best_first_graph_search with h_1	8.34	907	909	5,581	19	9.26	637	639	4,864	15
depth_limited_search						0.19	101	271	414	50
depth_first_graph_search	7.79	1,292	12,983	5,744	875	5.36	646	4,543	3,810	590
Problem Performance	464.25	5,146	7,266	38,412	136	186.27	2,438	3,109	18,378	84

When depth_limited_search found a search, it was typically the quickest to find a solution, but it was almost always insane. The best metric for measuring search performance (in my opinion) is **Optimal Plan Length** which averages the solutions to demonstrate performance across all 3 problems.

In general, the best search strategies found 9 nodes necessary; 6 for problem 1, 9 for problem 2, and 12 for problem 13. However some searches were more expensive than others, with astar_search_with_hg_pg_levelsum (Duration) and astar_search with h1 using the most time complexity, and breadth_first_search with space complexity (New Nodes, Expansions).

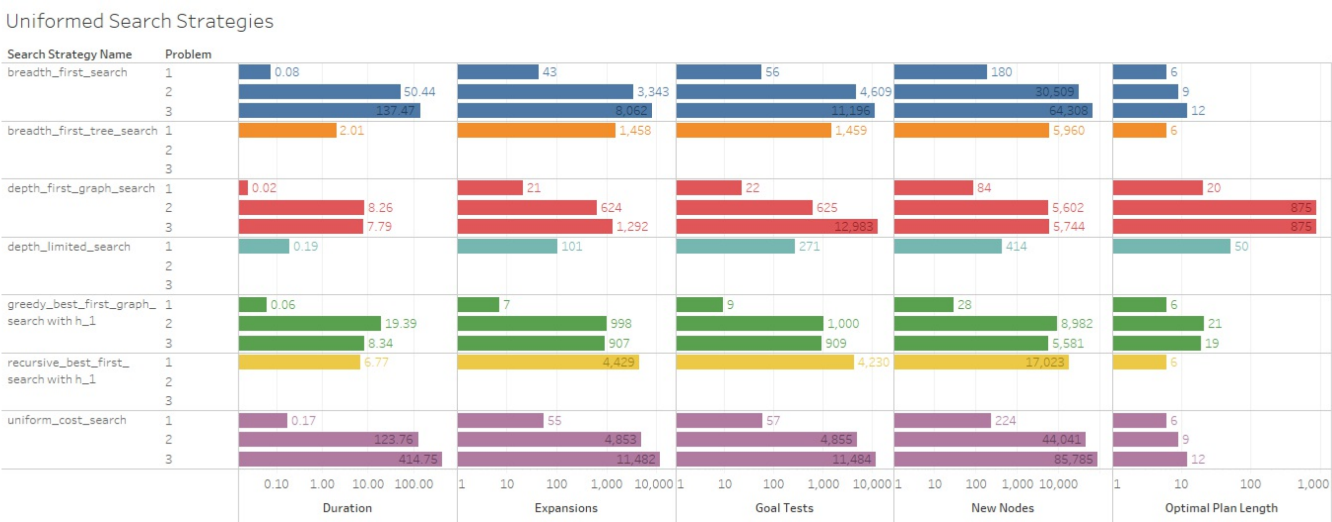
Optimal Plan

Problem 1	Problem 2	Problem 3
Load(C1,P1,		

SFO), Load(C2, P2, JFK), Fly(P1, SFO, JFK), Fly(P2, JFK, SFO), Unload(C1, P1, JFK), Unload(C2, P2, SFO)	Load(C3,P3, ATL), Fly(P3, ATL, SFO), Unload(C3, P3, SFO), Load(C2, P2, JFK), Fly(P2, JFK, SFO), Unload(C2, P2, SFO), Load(C1, P1, SFO), Fly(P1, SFO, JFK), Unload(C1, P1, JFK)	Load(C1,P1, SFO), Fly(P1, SFO, ATL), Load(C3, P1, ATL), Fly(P1, ATL, JFK), Unload(C3, P1, JFK), Unload(C1, P1, JFK), Load(C2, P1, JFK), Fly(P1, JFK, ORD), Load(C4, P1, ORD), Fly(P1, ORD, SFO), Unload(C4,P1, SFO), Unload(C2, P1, SFO)

These were the shortest search paths found by any kind of search, informed or otherwise. In testing, informed searches which leverage heuristics took longer to run, but astar_search with h_ignore_preconditions discovered an reasonable optimal path extraordinarily quickly, albeit expensively, finding reasonable solutions better than most uninformed strategies. However, the uninformed strategies like breadth_first_search, uniform_cost_search also discovered the optimal solution, also relatively expensively in both cases.

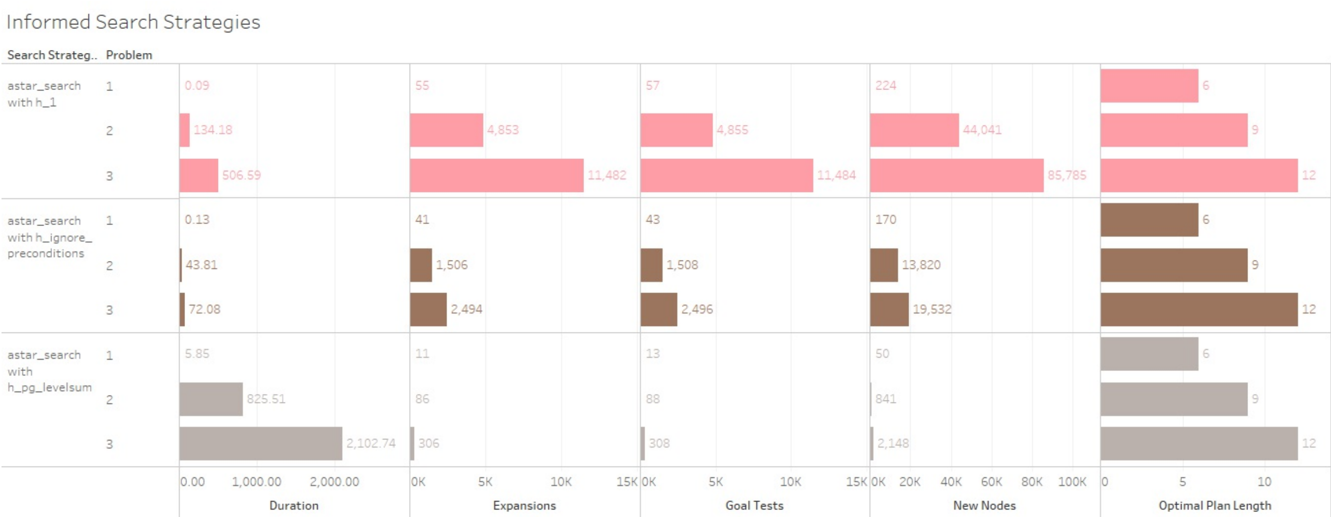
Uninformed Search Strategies



These strategies are uninformed, and rely only on the immediate frontier to make decisions.

- breadth_first_search: Reliable, sane, and optimal. Expensive; both in terms of time and space complexity. Durations well above the average, and far too many new nodes and goal state checks. It is comforing to know there at exists a structure that could attempt to find a solution to a crazy tree if given enough computational power.
- depth_first_graph_search: This typically yields some of the most hilarious up airpline itenaries I've ever seen, with an "optimal plan" of only 875 actions. At least it finds it quickly in small networks; in large networks it just seems to chase nonsense for infinity.
- uniform_cost_search: A variant of BFS but with a path cost to approximate with. Unfortunately in this case, it performs much worse than all the other search strategies, wasting time and resources. However, it still made it to the optimal solution, so at least there's that.
- greedy_best_first_graph_search: I really wish there was a way I could wrap my head around this; I'm just not getting it as well as the others. Do you guys have any suggestions?

Heuristic Search Strategies



Compare and contrast heuristic search result metrics using A* with the and "level-sum" heuristics for Problems 1, 2, and 3.

astar_search with h_pg_levelsum used the fewest the resources implying reduced algorithmic complexity. I was very impressed with its performance; but as the most well defined problem, this is not atypical. That said, all of the informed searches outranked the uninformed searches, always found the best solution, often with an increased cost in space (astar_search_with_h1) or time (astar_search_with_h_ignore_preconditions).

Conclusion

h_1	**h_ignore_preconditions**	**h_pg_levelsum**
# note that this is not a true heuristic This heuristic is just a constant of 1	This heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.	This heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

Summary Table

	Problem															Search Performance				
	1					2					3									
Search Strategy Name	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length
breadth_first_tree_search	2.01	1,458	1,459	5,960	6											2.01	1,458	1,459	5,960	6
recursive_best_first_search with h_1	6.77	4,429	4,230	17,023	6											6.77	4,429	4,230	17,023	6
astar_search with h_1	0.09	55	57	224	6	134.18	4,853	4,855	44,041	9	506.59	11,482	11,484	85,785	12	213.62	5,463	5,465	43,350	9
astar_search with h_ignore_preconditions	0.13	41	43	170	6	43.81	1,506	1,508	13,820	9	72.08	2,494	2,496	19,532	12	38.67	1,347	1,349	11,174	9
astar_search with h_pg_levelsum	5.85	11	13	50	6	825.51	86	88	841	9	2,102.74	306	308	2,148	12	978.03	134	136	1,013	9
breadth_first_search	0.08	43	56	180	6	50.44	3,343	4,609	30,509	9	137.47	8,062	11,196	64,308	12	62.66	3,816	5,287	31,666	9
uniform_cost_search	0.17	55	57	224	6	123.76	4,853	4,855	44,041	9	414.75	11,482	11,484	85,785	12	179.56	5,463	5,465	43,350	9
greedy_best_first_graph_search with h_1	0.06	7	9	28	6	19.39	998	1,000	8,982	21	8.34	907	909	5,581	19	9.26	637	639	4,864	15
depth_limited_search	0.19	101	271	414	50											0.19	101	271	414	50
depth_first_graph_search	0.02	21	22	84	20	8.26	624	625	5,602	875	7.79	1,292	12,983	5,744	875	5.36	646	4,543	3,810	590
Problem Performance	1.54	622	622	2,436	12	172.19	2,323	2,506	21,119	134	464.25	5,146	7,266	38,412	136	186.27	2,438	3,109	18,378	84

astar_search with h_pg_levelsum

Relatively more space effecient than any other search strategy, despite taking a little longer to run. With the entire structure of the planning graph, it is the most well defined problem.

astar_search_with_h1

Wasn't really an informed search since h1 is arbitrary, so it ended up just being a more expensive BFS, since we took one step further than we needed to every time.

astar_search_with_h_ignore_preconditions

Yielded some significant savings as well, and reached an optimal solution faster without the whole planning graph. Definitely the all-star in my book; a good tradeoff between implementational scale and effeciency.

Summary Table

Problem

1

Search Strategy Name	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length
breadth_first_tree_search	2.01	1,458	1,459	5,960	6
recursive_best_first_search with h_1	6.77	4,429	4,230	17,023	6
astar_search with h_1	0.09	55	57	224	6
astar_search with h_ignore_preconditions	0.13	41	43	170	6
astar_search with h_pg_levelsum	5.85	11	13	50	6
breadth_first_search	0.08	43	56	180	6
uniform_cost_search	0.17	55	57	224	6
greedy_best_first_graph_search with h_1	0.06	7	9	28	6
depth_limited_search	0.19	101	271	414	50
depth_first_graph_search	0.02	21	22	84	20
Problem Performance	1.54	622	622	2,436	12

2

Search Strategy Name	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length
breadth_first_tree_search					
recursive_best_first_search with h_1					
astar_search with h_1	134.18	4,853	4,855	44,041	9
astar_search with h_ignore_preconditions	43.81	1,506	1,508	13,820	9
astar_search with h_pg_levelsum	825.51	86	88	841	9
breadth_first_search	50.44	3,343	4,609	30,509	9
uniform_cost_search	123.76	4,853	4,855	44,041	9
greedy_best_first_graph_search with h_1	19.39	998	1,000	8,982	21
depth_limited_search					
depth_first_graph_search	8.26	624	625	5,602	875
Problem Performance	172.19	2,323	2,506	21,119	134

3

Search Strategy Name	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length
breadth_first_tree_search					
recursive_best_first_search with h_1					
astar_search with h_1	506.59	11,482	11,484	85,785	12
astar_search with h_ignore_preconditions	72.08	2,494	2,496	19,532	12
astar_search with h_pg_levelsum	2,102.74	306	308	2,148	12
breadth_first_search	137.47	8,062	11,196	64,308	12
uniform_cost_search	414.75	11,482	11,484	85,785	12
greedy_best_first_graph_search with h_1	8.34	907	909	5,581	19
depth_limited_search					
depth_first_graph_search	7.79	1,292	12,983	5,744	875
Problem Performance	464.25	5,146	7,266	38,412	136

Search Performance

Search Strategy Name	Duration	Expansions	Goal Tests	New Nodes	Optimal Plan Length
breadth_first_tree_search	2.01	1,458	1,459	5,960	6
recursive_best_first_search with h_1	6.77	4,429	4,230	17,023	6
astar_search with h_1	213.62	5,463	5,465	43,350	9
astar_search with h_ignore_preconditions	38.67	1,347	1,349	11,174	9
astar_search with h_pg_levelsum	978.03	134	136	1,013	9
breadth_first_search	62.66	3,816	5,287	31,666	9
uniform_cost_search	179.56	5,463	5,465	43,350	9
greedy_best_first_graph_search with h_1	9.26	637	639	4,864	15
depth_limited_search	0.19	101	271	414	50
depth_first_graph_search	5.36	646	4,543	3,810	590
Problem Performance	186.27	2,438	3,109	18,378	84