Heuristic Analysis

Air Cargo Analysis

Problem 1:

	Expansions	Goal Tests	New Nodes	Plan Length	Time
breadth_first_search	43	56	180	6	0.4149
depth_first_graph_search	22	22	84	20	0.2067
uniform_cost_search	55	57	224	6	0.5115
astar_search h_ignore_preconditions	41	43	170	6	0.5067
astar_search h_pg_levelsum	11	13	50	6	0.5514

Problem 2:

	Expansions	Goal Tests	New Nodes	Plan Length	Time
breadth_first_search	3343	4609	30509	9	216.9260
depth_first_graph_search	624	625	5602	619	42.3216
uniform_cost_search	4852	4854	44030	9	311.0809
astar_search h_ignore_preconditions	1450	1452	13303	9	109.9737
astar_search h_pg_levelsum	86	88	841	9	39.3848

Problem 3:

	Expansions	Goal Tests	New Nodes	Plan Length	Time
breadth_first_search	14663	18098	129631	12	1330.3625
depth_first_graph_search	408	409	3364	392	29.5769
uniform_cost_search	18235	18237	159716	12	1481.7209
astar_search h_ignore_preconditions	5040	5042	44944	12	519.1684
astar_search h_pg_levelsum	315	317	2902	12	222.2085

= Optimal plan

Non-Heuristic Search Comparison:

I arbitrarily chose the uniform cost search for my third non-heuristic search. Of the three searches, breadth-first search consistently performed the best, with uniform-search coming in 2nd, and depth-first graph search coming in dead last due to the size of the solution plan lengths. Time-wise, depth-first graph search performed well, but its plan lengths rendered its solutions unusable in a real-world situation.

This makes sense, as while breadth-first and uniform-cost searches are optimal, depth-first search is not, so we cannot rely on its plan length to be the most optimal it can be. I was surprised, however, by the magnitude of difference we see between these plans. It also makes sense that depth-first would be the search that executes the quickest. While breadth-first and

uniform-cost searches traverse the whole graph, depth-first does not, resulting in a much smaller explored set, and quicker execution times.

It also makes sense that breadth-first and uniform-cost searches end up with a similar number of explored nodes, with uniform-cost being slightly higher. Since uniform-cost search has a more variable frontier at any given state, the possibility of overextending its search past the scope of breadth-first search is much higher.

I can draw one main, glaring conclusion from this: Only use depth-first search if you don't care about the optimality of the solution. It's quick and it gets the job done, just in a potentially extremely un-efficient manner, in terms of the solution. It's more efficient than breadth-first and uniform-cost, if all we care about is the amount of processing time and space used.

Heuristic Search Comparison:

Between the ignore-preconditions and level-sum heuristics, level-sum consistently performed better in each of the three problems – both in time and space.

Seeing as how both of these heuristics are estimations, I really had no idea how each would perform going into the project. So, it was surprising seeing the large divide in performance between the two. One thing I'm a bit worried about is how I implemented the ignore-preconditions heuristic – I wonder if there was a more optimal way to develop it than the solution I coded.

Intuitively, it's difficult for me to see when the ignore-preconditions heuristic could be an effective heuristic to use. Since it does not factor in the negative effects of the action, it makes sense that it could go down many inefficient paths, not realizing that the next action is actually moving us away from the goal.

Conclusions:

The biggest conclusions I've drawn from this project are:

1. The type of search matters immensely, and should be chosen based on each individual planning problem.
2. The heuristic used matters just as much, and should also be chosen based on each individual planning problem.