Planning Search – Written Analysis

In Figure 1 below, all of the planning algorithms were executed for all three of the cargo problems. Those line items with no numbers and filled with red were items whose execution time was "excessive" to the point of not being worth considering/discussing. Green rows indicate those algorithms with the best execution time for a given problem.

With regard to the recommended algorithm for each problem, for problems 1 and 3 depth first provides the best performance in terms of execution time; for problem 2, greedy best first graph search with h_1 is recommended based on execution time. Overall this was surprising to me that depth first performed as well as it did for all three of the problem. I tend to view achieving good results with depth first as "getting lucky" or that the algorithm is best-suited for very specific types of problems where the goal frequently occurs on the left side of the tree being searched to avoid excessive execution times as depth first is not "complete" so there is no guarantee of finding a goal (for example, if the left-most branch is infinite – as discussed in earlier lectures). What is also interesting is that depth first is the fastest algorithm for problem 3 despite having the largest plan length – this is likely due to the fact that it also has the lowest number of node expansions.

The A* with the levelsum heuristic performed the worst of all algorithms that incorporated heuristics while greedy-first with h1 performed the best; A* with h1 and ignore pre-conditions had similar execution times and exactly the same plan lengths/expansions/goal tests/new nodes and came in second place. The levelsum's performance hit comes mainly from the number of nested loops for determining the level cost for each of the goals whereas ignoring preconditions simplifies the problem that needs to be solve consequently reducing the number of conditions that need to be checked.

Overall heuristics did not help the performance of these algorithms when compared to the non-heuristic algorithms; as discussed, the levelsum added a huge amount of overhead in terms for performance, the h1 heuristic isn't really a heuristic (per the comment in the function in the code and the fact that it just returns 1), and even ignoring preconditions did not make the A* algorithms competitive compared to the other non-heuristic algorithms.

Figure 1

Problem	Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed
Cargo Problem 1	breadth_first_search	43	56	180	6	0.033
	breadth_first_tree_search	1458	1459	5960	6	1.034
	depth_first_graph_search	12	13	48	12	0.008
	depth_limited_search	101	271	414	50	0.095
	uniform_cost_search	55	57	224	6	0.039
	recursive_best_first_search with h_1	4229	4230	17029	3	3.033
	greedy_best_first_graph_search with h_1	7	9	28	6	0.005
	astar_search with h_1	55	57	224	6	0.04
	h_ignore_preconditions	55	57	224	6	0.043
	astar_search with h_pg_levelsum	45	47	188	6	1.196
Cargo Problem 2	breadth_first_search	3351	4572	29722	9	14.644
	breadth_first_tree_search					
	depth_first_graph_search	1609	1610	13478	637	6.436
	depth_limited_search					
	uniform_cost_search	4793	4795	42696	9	12.578
	recursive_best_first_search with h_1					
	greedy_best_first_graph_search with h_1	950	952	8350	13	2.492
	astar_search with h_1	4793	4795	42696	9	13.041
	h_ignore_preconditions	4793	4795	42696	9	12.934
	astar_search with h_pg_levelsum	1962	1964	17736	9	513.216
Cargo Problem 3	breadth_first_search	14663	18098	129631	12	112.049
	breadth_first_tree_search					
	depth_first_graph_search	592	593	4927	571	3.183
	depth_limited_search					
	uniform_cost_search	18235	18237	159716	12	57.925
	recursive_best_first_search with h_1					
	greedy_best_first_graph_search with h_1	5614	5616	49429	22	18.034
	astar_search with h_1	18235	18237	159716	12	57.645
	h_ignore_preconditions	18235	18237	159716	12	58.335
	astar_search with h_pg_levelsum	2841	2843	27040	12	1607.079