Project 2: Build a Forward Planning Agent

Experimental Results & Report

By: Sarp Karamarti

As shown in Table 1, the four reviewed air cargo problems differ in complexity with problem 1 having 20 actions in its domain, problem 2 having 72 actions in its domain etc... Taking a look at Table 2, we can see that the number of expanded nodes growths exponentially for all used algorithms. For the greedy algorithms the exponential behavior is not as obvious as for the others, however.

Table 1 - Number of actions in domain for each air cargo problem

	Problem 1	Problem 2	Problem 3	Problem 4
# of Actions in Domain	20	72	88	104

All air cargo problems have been solved with the below listed algorithms on a local machine and pypy3. As shown in Table 2, the harder the problem gets the longer the computation time. Here, we can again observe an exponential behavior for the majority of examined algorithms. Solving problem 4 required often a multiple of the computation time that was required to solve problem 3. An exception here is the greedy_best_first_graph_seach_h_unmet_goals algorithm.

Looking at the plan lengths we see that the depth first search algorithm found solutions with significantly longer plans than the other algorithms, with the multiple getting bigger with the complexity of the problem. What's interesting here is that the breadth first search algorithm found a solution with minimal plan length quite fast compared to the a* search algorithms with sum level, max level and set level heuristics. The discrepancy seems to only grow with problem complexity (e.g. a* with maxlevel heuristic takes for problem 3 only 274 times as long as breadth first search but 390 times longer for problem 4).

In very restrictive domains that have to operate in real time a depth first search algorithm seems appropriate. For a small, finite domain depth first search always finds a solution and does this very quickly too. Same goes for the greedy best first graph search algorithms, with the unmet goals and level sum heuristics being the fastest.

Although in this experiment not as performed as breadth first search and uniform cost search, best first search and A* search could be a better choice for very large domains (if there is information to build a good heuristic) as the heuristic can help to get faster to a solution. If there is no such information available, breadth first search and uniform cost will also find a solution.

If the objective is to find only optimal plans, breadth first search and uniform cost (when cost function is different than plan length) search are recommended.

Table 2 - Number of expansions, plan lengths and computation time for problems and algorithms

Methods	Expansions				Plan Length				Time [seconds]			
	P1	**P2**	**P3**	**P4**	**P1**	**P2**	**P3**	**P4**	**P1**	**P2**	**P3**	**P4**
breadth_first_search	43	3343	14663	99736	6	9	12	14	0,019	0,314	0,674	4,845
depth_first_graph_search	21	624	408	25174	20	619	392	24132	0,006	0,487	0,156	692,852
uniform_cost_search	60	5154	18510	113339	6	9	12	14	0,015	0,579	1,020	7,351
greedy_best_first_graph_search h_unmet_goals	7	17	25	29	6	9	15	18	0,002	0,017	0,012	0,008
greedy_best_first_graph_search h_pg_levelsum	6	9	14	17	6	9	14	17	0,290	0,262	0,559	0,848
greedy_best_first_graph_search h_pg_maxlevel	6	27	21	56	6	9	13	17	0,122	0,422	0,535	1,553
greedy_best_first_graph_search h_pg_setlevel	6	9	35	107	6	9	17	23	0,533	0,962	4,185	18,733
astar_search h_unmet_goals	50	2467	7388	34330	6	9	12	14	0,011	0,440	0,694	3,542
astar_search h_pg_levelsum	28	357	369	1208	6	9	12	15	0,254	5,702	10,284	55,116
astar_search h_pg_maxlevel	43	2887	9580	62077	6	9	12	14	0,130	30,867	184,705	1893,704
astar_search h_pg_setlevel	33	1037	3423	22606	6	9	12	14	0,325	68,745	365,055	3961,644