Planning Search Heuristic Analysis

Using an algorithm guaranteeing to find an optimal solution if it exists, like breadth-first search, we can ensure there is not better solution, for the problems proposed, than the ones showed below (actions taking part in the goal achievement are shown in bold for validation purpose).

Problem	Optimal path length	Optimal path
1	6	Load(C1, P1, SFO) → Load(C2, P2, JFK) → Fly(P2, JFK, SFO) → Unload(C2, P2, SFO) → Fly(P1, SFO, JFK) → **Unload(C1, P1, JFK)**
2	9	Load(C1, P1, SFO) → Load(C2, P2, JFK) → Load(C3, P3, ATL) → Fly(P2, JFK, SFO) → **Unload(C2, P2, SFO)** → Fly(P1, SFO, JFK) → **Unload(C1, P1, JFK)** → Fly(P3, ATL, SFO) → **Unload(C3, P3, SFO)**
3	12	Load(C1, P1, SFO) → Load(C2, P2, JFK) → Fly(P2, JFK, ORD) → Load(C4, P2, ORD) → Fly(P1, SFO, ATL) → Load(C3, P1, ATL) → Fly(P1, ATL, JFK) → **Unload(C1, P1, JFK)** → **Unload(C3, P1, JFK)** → Fly(P2, ORD, SFO) → **Unload(C2, P2, SFO)** → **Unload(C4, P2, SFO)**

Table 1: Optimal path for each problem.

Results

Even though there isn't exist a better solution, it is a fact that some algorithms find it in quite less time than others. Next, it is shown a comparative table pointing out the differences between some search strategies for each one of the proposed problems.

Problem 1				
Search Strategy	Path Length	Is optimal	Time (s)	Node Expansions
breadth_first_search	6	True	0.0231	43
breadth_first_tree_search	6	True	0.5982	1458
depth_first_graph_search	20	False	0.0107	21
depth_limited_search	50	False	0.0645	101
uniform_cost_search	6	True	0.0267	55
recursive_best_first_search h_1	6	True	1.7604	4229
greedy_best_first_graph_search h_1	6	True	0.0044	7
astar_search h_1	6	True	0.0275	55
astar_search h_ignore_preconditions	6	True	0.0220	41
astar_search h_pg_levelsum	6	True	0.4247	11

Table 2: Problem 1 executions

Problem 2				
Search Strategy	Path Length	Is optimal	Time (s)	Node Expansions
breadth_first_search	9	True	11.2593	3343
breadth_first_tree_search	-	-	-	-
depth_first_graph_search	619	False	2.7545	624
depth_limited_search	50	False	748.7611	222719
uniform_cost_search	9	True	9.4070	4853
recursive_best_first_search h_1	-	-	-	-
greedy_best_first_graph_search h_1	17	False	1.9259	998
astar_search h_1	9	True	9.4789	4853
astar_search h_ignore_preconditions	9	True	3.1700	1450
astar_search h_pg_levelsum	9	True	36.2409	86

Table 3: Problem 2 executions

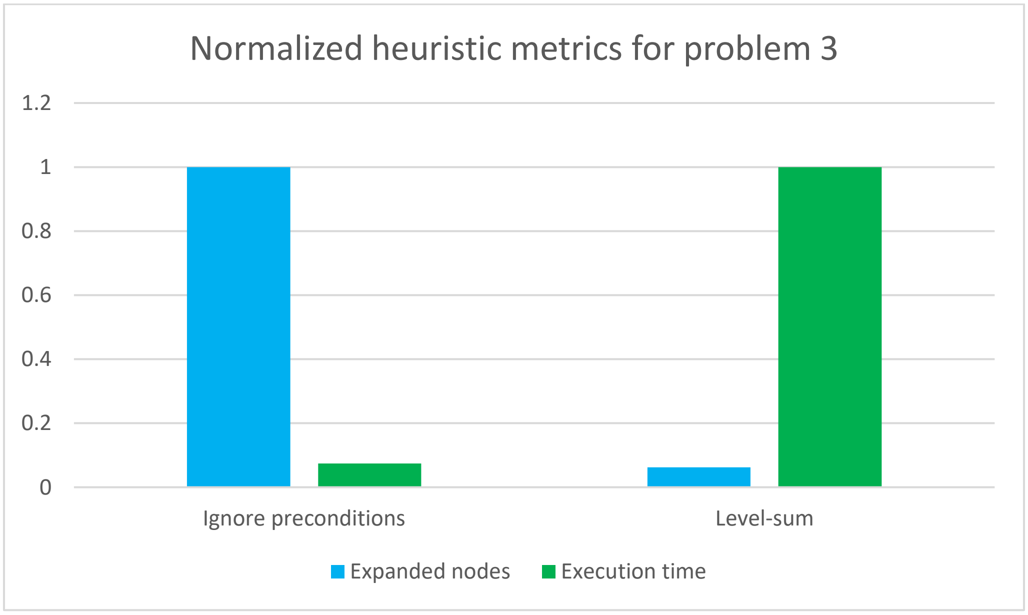
Problem 3				
Search Strategy	Path Length	Is optimal	Time (s)	Node Expansions
breadth_first_search	12	True	81.3276	14663
breadth_first_tree_search	-	-	-	-
depth_first_graph_search	392	False	1.4962	408
depth_limited_search	-	-	-	-
uniform_cost_search	12	True	43.04	18164
recursive_best_first_search h_1	-	-	-	-
greedy_best_first_graph_search h_1	26	False	12.4474	5398
astar_search h_1	12	True	42.3205	18164
astar_search h_ignore_preconditions	12	True	12.9561	5038
astar_search h_pg_levelsum	12	True	174.3222	314

Table 4: Problem 3 executions

Analysis

Making a comparison between the uninformed non-heuristic search strategies, we can observe than only two of them were capable of finding an optimal solution, these were BFS (Breadth-First Search) and UCS (Uniform Cost Search). We can notice that UCS performs better than BFS as the search-space increases which is strange given that BFS expands less nodes than UCS, but after a quick research I found it is due to an optimization in the data structure implementation that UCS uses. In the other hand, it is easy to see that DFS (Depth-First Search) algorithms fails in finding an optimal solution since they go as deep as they can in a branch before looking in another one, and setting a depth limit does not help, in fact the algorithm becomes slower at finding a solution.

Regarding to heuristic search we can observe that just A* succeed in finding an optimal solution, being the "ignore preconditions" heuristic the fastest. Here it is important to notice that "h_1" heuristic is not a true heuristic as it returns 1 all the time no matter anything, this means that A*, using this heuristic, behaves as UCS and indeed we can observe the similitude in nodes expanded and running time between them.



If we compare "ignore preconditions" (H2) and "level-sum" (H3) heuristics we can realize that the first one outperforms the second one when it comes to run time but it gets reversed when we look at the number of nodes expanded. The explanation is quite simple, H3 expands fewer nodes because it gets better estimations however the time to calculate them is big since it uses a complex algorithm called GraphPlan (basically another search algorithm). On the other hand H2 gets no-so-good estimations as H3 but it gets them in less time by defining a relaxed problem that is easier to solve.

Conclusion

In conclusion heuristic-search strategies are better than uninformed non-heuristic strategies as search-space increases. This is because heuristic search focuses on those nodes which have better odds to lead to goal state according to the estimations given by some heuristic. Moreover a good heuristic is a tradeoff between having good estimations and the time taken to calculate such estimations. It is preferable to have an heuristic with no so good estimations like H2 than an heuristic with better estimations like H3 but that takes longer in being calculated.