Planning Search Algorithms Analysis

Metrics for non-heuristic planning solution searches

air_cargo_p1

Algorithm	Node expansions	Goal tests	New nodes	Plan length	Time elapsed (in seconds)	Optimality
Breath first search	43	56	180	6	0.035	Yes
Depth first graph search	21	22	84	20	0.01	No
Uniform cost search	55	57	224	6	0.05	Yes

air_cargo_p2

Algorithm	Node expansions	Goal tests	New nodes	Plan length	Time elapsed (in seconds)	Optimality
Breath first search	3346	4612	30534	9	9.00	Yes
Depth first graph search	107	108	959	105	0.33	No
Uniform cost search	4853	4855	44041	9	12.77	Yes

air_cargo_p3

Algorithm	Node expansions	Goal tests	New nodes	Plan length	Time elapsed (in seconds)	Optimality
Breath first search	14120	17673	124926	12	43.10	Yes
Depth first graph search	292	293	2388	288	1.15	No
Uniform cost search	18233	18235	159697	12	52.04	Yes

Overall Analysis

It can be seen from the tables above, Breath first search(BFS) and Uniform cost search always returned optimal sequence of actions for each problem. This is due to the nature of the algorithms that expand to neighbors of the current node first, but Uniform cost search prefers nodes with lower path cost. In this part however, heuristics are not applied, so Uniform cost search has no knowledge of the nodes to expand next giving no advantage over BFS. One point to note here is the number of node expansions, goal tests new nodes and time elapsed. BFS always gave lower values as this implementation of BFS terminates early as soon as the goal state is added to the frontier whereas, Uniform cost search waits until the goal state is popped out of the frontier and then terminates. Both BFS and Uniform cost search guarantee minimum number of steps to the goal.

Depth first search (DFS) on the other hand gave non-optimal sequence of actions for each problem because it expands deep in the tree first before checking if the goal is achieved. This doesn’t guarantee minimum number of steps to the goal and even worse never reaches the goal if the number of states is infinite. However, DFS has a huge advantage in terms of the space saved as it doesn’t keep track of the explored set. In terms of search time, DFS achieved the goal faster compared to the other algorithms because all neighboring nodes of the current node are not processed before expanding to the next level of the tree.

Peeranat Fupongsiripan

Metrics of A* searches with heuristics

air_cargo_p1

Algorithm	Node expansions	Goal tests	New Nodes	Plan length	Time elapsed (in seconds)	Optimality
A* with h1	55	57	224	6	0.05	Yes
A* with h_ignore_preconditions	41	43	170	6	0.05	Yes
A* with h_pg_levelsum	11	13	50	6	0.53	Yes

air_cargo_p2

Algorithm	Node expansions	Goal tests	New Nodes	Plan length	Time elapsed (in seconds)	Optimality
A* with h1	4853	4855	44041	9	11.87	Yes
A* with h_ignore_preconditions	1450	1452	13303	9	4.19	Yes
A* with h_pg_levelsum	86	88	841	9	44.86	Yes

air_cargo_p3

Algorithm	Node expansions	Goal tests	New nodes	Plan length	Time elapsed (in seconds)	Optimality
A* with h1	18233	18235	159697	12	51.83	Yes
A* with h_ignore_preconditions	4951	4953	44051	12	16.21	Yes
A* with h_pg_levelsum	316	318	2926	12	236.50	Yes

Overall Analysis

Overall, it is obvious from the tables above that A* search with heuristics returned optimal sequence of actions. This is because hints to the goal are provided to the A* algorithm. However, this only holds true if the heuristic is admissible(optimistic, or not overestimate). Heuristic h1 always returns 1 and therefore behaves as uniform cost search. In contrast, heuristic h_ignore_preconditions is actually the minimum number of steps to the goal which is always greater than or equal to h1 therefore, will expand fewer paths than h1. This can be seen in the numbers of node expansions, goal tests, new nodes, and time elapsed which are always lower than those of h1,

A* with h_pg_levelsum is much worse than others in terms of search time because a planning graph is built in polynomial time. For this reason, h_pg_levelsum can be an overkill if search time is crucial to our problem. It is also evident h_pg_levelsum expanded less paths compared to its counterparts as seen in node expansions, goal tests, and new nodes. Furthermore, this heuristic isn't guaranteed to return optimal solutions as it is inadmissible and doesn't perform well especially when the problem is largely not decomposable, even though in these air cargo problems it returned an optimal sequence of actions.

At the end of the day, considering the results of all search algorithms with and without heuristics functions, heuristics always help the algorithms find shortest path to the goal provided that the heuristics are admissible. It may also cause faster search time to find solutions if the heuristic is not too complex. There's a trade-off between fewer paths to expand and fast search time. Reducing the number of paths to expand often comes at a cost of constructing highly accurate heuristics which is often expensive to calculate (e.g. Planning Graph), resulting in slower search time. In these air cargo problems in particular, the best heuristic is h_ignore_preconditions as it's admissible, guaranteed to return optimal solutions and executed fastest even though it may cause A* to expand more nodes than h_pg_levelsum. In reality, optimal plan for cargo shipment is the most important aspect as the cost of shipment is huge.

Peeranat Fupongsiripan