Planning Search Heuristic Analysis

Optimal plan

Problem 1 (6 steps)	Problem 2 (9 steps)	Problem 3 (12 steps)

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
	Load(C3, P3, ATL)
	Fly(P3, ATL, SFO)
	Unload(C3, P3, SFO)
	Load(C1, P1, SFO)
	Fly(P1, SFO, JFK)
	Unload(C1, P1, JFK)
	Load(C2, P2, JFK)
	Fly(P2, JFK, SFO)
	Unload(C2, P2, SFO)
		Load(C2, P2, JFK)
		Fly(P2, JFK, ORD)
		Load(C4, P2, ORD)
		Fly(P2, ORD, SFO)
		Unload(C4, P2, SFO)
		Load(C1, P1, SFO)
		Fly(P1, SFO, ATL)
		Load(C3, P1, ATL)
		Fly(P1, ATL, JFK)
		Unload(C3, P1, JFK)
		Unload(C1, P1, JFK)
		Unload(C2, P2, SFO)

Optimality of an algorithm:

Before we go into comparing the non-heuristic algorithms with heuristic ones, it is important to define optimality of an algorithm. I think in this use case; the optimality of the plan is to find minimum possible length of plan even though the algorithm takes more time or expands more nodes than other algorithms because computing time is cheap compared to 2 more useless trips from one city to another.

Breadth First Search (BFS):

Because the goal is to find minimum edges to reach our goal, naturally BFS should do better than other algorithms for this use case. Although I found that as the problem got bigger the time BFS is spending to find the optimum path is growing very rapidly.

Depth First Graph Search (DFGS):

For this problem, depth first search is a very bad choice because there are so many combinations and traversing the depth of the tree to find a solution will result in a very long path, which is what I found when I ran depth first graph search for the three problems.

Uniform Cost Search (UCS):

While uniform cost search guarantees the optimum solution for this problem, it is very compute intensive because it is going through all the paths and then finding the minimum cost path that satisfies the problem

A* with planning graph level sum heuristic:

I expected this algorithm to perform best but I found out that it takes a long time for this algorithm to finish, after waiting more than 15 minutes for air cargo problem 2, I gave up on this and moved to the ignore preconditions heuristic.

A* with ignore preconditions heuristic:

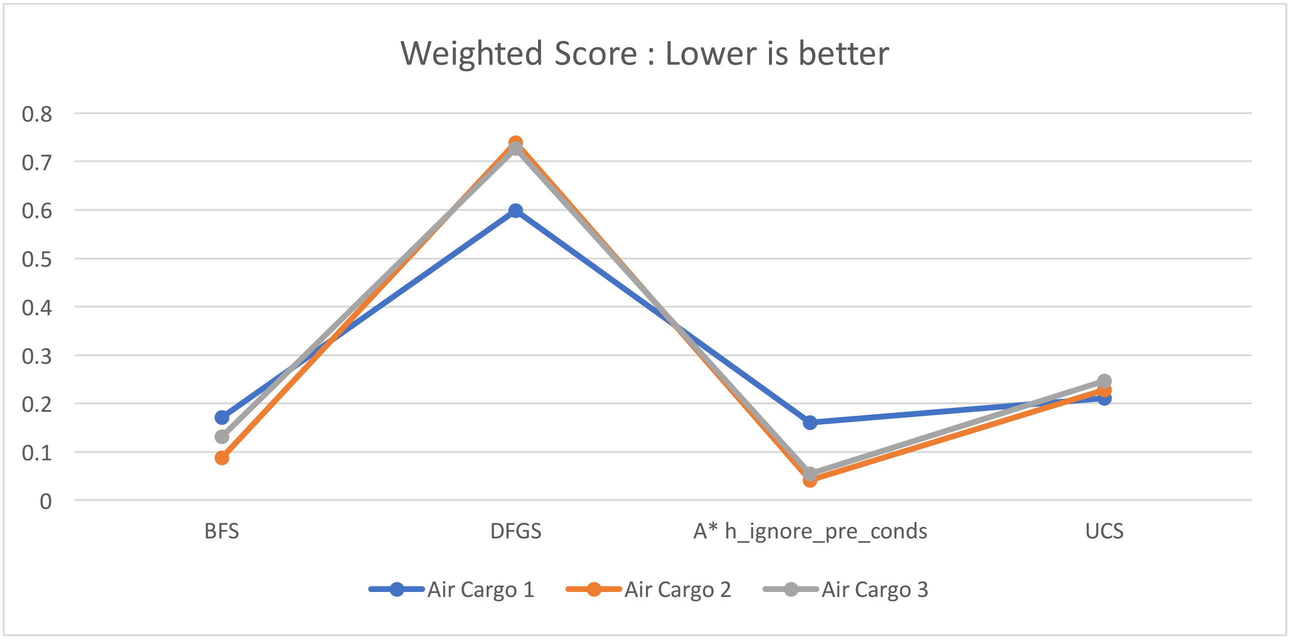
This is the best algorithm for the given problem and as shown in results below, it gives optimum plan in least amount of time for all 3 problems (except for greedy first search which is better for problem 1 but it performs so badly for problem 2 and 3 that it is not a good candidate to mention)

Problem	BFS Plan Length	DFGS Plan Length	A* h_ignore_pre_cond	UCS plan length
Air Cargo 1	6	20	6	6
Air Cargo 2	9	619	9	9
Air Cargo 3	12	392	12	12

Problem	BFS Time	DFGS Time	A* h_ignore_pre_cond Time	UCS Time
Air Cargo 1	0.035	0.02	0.033	0.039
Air Cargo 2	14.43	3.5	15.08	74.25
Air Cargo 3	128.54	1.88	88	465.26

Problem	BFS Expansions	DFGS Exp	A* h_ignore_pre_cond Expans	UCS Expans
Air Cargo 1	43	21	41	55
Air Cargo 2	3343	624	1506	4853
Air Cargo 3	14663	408	5118	18221

To make it easier to understand and compare, I have created a chart with weighted optimality scores with normalized values for plan length, time taken, and nodes expanded for all three problems and 4 algorithms. The normalizing and weighting math is in the excel sheet “weighted_metrics” which is also provided with the assignment.


Weighted Score : Lower is better
BFS
DFGS
A* h_ignore_pre_conds
UCS
Air Cargo 1
Air Cargo 2
Air Cargo 3