Heuristic Analysis

I. Problems and solutions

For the provided problems, I devised the following optimal solutions:

1.1 Problem 1 initial state and goal

Init(At(C1, SFO) ∧ At(C2, JFK)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))

1.2 Problem 1 Solution – Length : 6

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

2.1 Problem 2 initial state and goal

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

2.2 Problem 2 Solution – Length : 9

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

3.1 Problem 3 initial state and goal

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

3.2 Problem 3 Solution – Length : 12

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

II. Analysis of Uninformed and Heuristic Searches:

In the following table, results from running various uninformed and a-star heuristic searches on are provided.

	Search	Problem	Plan Length	Optimal	New Nodes	Goal Tests	Expansions	Time in Seconds
Uninformed	Breadth First Search	1	6	YES	180	56	43	0,0265
		2	9	YES	30509	4609	3343	11,9085
		3	12	YES	129631	18098	14663	87,9806
	Depth First Graph Search	1	12	NO	48	13	12	0,0069
		2	575	NO	5211	583	582	2,7262
		3	596	NO	5176	628	627	2,8143
	Greedy Best First Graph Search	1	6	YES	28	9	7	0,0041
		2	17	NO	8910	992	990	2,1275
		3	23	NO	49285	5598	5596	14,1962
A* Heuristic	Ignore Preconditions	1	6	YES	170	43	41	0,0309
		2	9	YES	13303	1452	1450	3,7884
		3	12	YES	44422	4990	4988	14,7832
	Level Sum	1	6	YES	50	13	11	0,7058
		2	9	YES	841	88	86	135,0773
		3	12	YES	2722	298	296	739,9194

1. Uninformed Search Comparison

BFS is the only non-heuristic search, which always finds the optimal solution, and thus it would be the obvious choice if an optimal solution were required. This was to be expected as we can see in the “Artificial Intelligence a Modern Approach” book, which says:

“We can easily see that it is *complete*—if the shallowest goal node is at some finite depth *d*, breadth-first search will eventually find it after generating all shallower nodes (provide the branching factor b is finite). Note that as soon as a goal node is generated, we know it is the

shallowest goal node because all shallower nodes must have been generated already and failed the goal test".[1]

However, it does much more node expansions and goal tests than DFS and, as a result, uses a lot more memory. It also takes much longer time to finish, so if an optimal length were not required, DFS would be a better choice.

Greedy Best First Graph Search is faster and uses less memory than both BFS and DFS for simpler problems and as problems get harder it takes the middle ground between them. It would be a good choice if we needed a solution with length that is not necessarily optimal, but is not too far off from it, as it could provide such a solution faster than BFS and with less memory used.

2. Heuristic Search Comparison

Both heuristic functions provide optimal problem solutions for all three problems. The main differences are in the time and memory used to accomplish this. The Level Sum function takes much longer to finish (probably because it needs to construct a Planning Graph and make all the mutex checks for all states and actions), but uses much less memory than the Ignore Preconditions function.

3. Uninformed and Heuristic Search Comparison

If an optimal solution was needed then an A star search with the Ignore Preconditions heuristic would definitely be a very good choice – it runs faster than BFS and takes less memory.

If memory used was an issue then an A star search with the Level Sum function would be a perfect pick – it has a lower number of expansions and goal tests than all uniformed search methods.

If we were mostly concerned about the time used then DFS would be the right choice, because not only does it run very fast, but also the time needed for all other searches appears to increase exponentially, while the increase in time from problem 2 to problem 3 for DFS is just from 2.7 to 2.8 seconds.

[1] Artificial Intelligence a Modern Approach 3rd Edition, Stuart J. Russell and Peter Norvig