Heuristic analysis

Problem 1 search results and optimal path:

No	Search	Expansions	Goal Tests	New Nodes	Plan length	Time taken [s]	Optimal solution
1	BFS	43	56	180	6	0.04	YES
2	Breadth First Tree	1458	1459	5960	6	1.21	YES
3	DFS	21	22	84	20	0.02	NO
4	Depth Limited	101	271	414	50	0.11	NO
5	Uniform Cost	55	57	224	6	0.05	YES
6	Recursive BFS	4229	4230	17023	6	3.56	YES
7	Greedy Best First	7	9	28	6	0.01	YES
8	A*	55	57	224	6	0.05	YES
9	A* ignore precond.	41	43	170	6	0.05	YES
10	A* levelsum	11	13	50	6	1.04	YES

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

For problem 1, the quickest solution was a greedy best first graph search. While not only being the quickest, it was also providing an optimal path. Slowest of all searches was the recursive BFS. All searches managed to arrive at an answer in reasonable time. The depth limited search was allowed only a relatively small recursion limit and arrived at the longest plan length.
BFS and DFS both have a reasonable runtime, but DFS does not find an optimal solution.

Problem 2 search results and optimal path:

No	Search	Expansions	Goal Tests	New Nodes	Plan length	Time taken [s]	Optimal solution
1	BFS	3343	4609	30509	9	14.86	YES
2	Breadth First Tree	-	-	-	-	TIMEOUT	-
3	DFS	624	625	5602	619	3.50	NO
4	Depth Limited	-	-	-	-	TIMEOUT	-
5	Uniform Cost	4853	4855	44041	9	14.14	YES
6	Recursive BFS	-	-	-	-	TIMEOUT	-
7	Greedy Best First	998	1000	8982	15	2.84	NO
8	A*	4853	4855	44041	9	13.89	YES
9	A* ignore precond.	1450	1452	13303	9	4.99	YES
10	A* levelsum	86	88	841	9	185.21	YES

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

Again, greedy best first search took the crown, being the fastest, on this search. However, the solution is not optimal any more. Search time are starting to become an issue as the complexity increases. Resource intensive searches are timing out now. The complexity leads to more nodes expanded as well. BFS found an optimal solution, but took a comparatively long time to complete. On the other hand, DFS was among the quickest searches, but is far away from the optimal path.

Problem 3 search results and optimal path:

No	Search	Expansions	Goal Tests	New Nodes	Plan length	Time taken [s]	Optimal solution
1	BFS	14663	18098	129631	12	107.30	YES
2	Breadth First Tree	-	-	-	-	TIMEOUT	-
3	DFS	408	409	3364	392	1.95	NO
4	Depth Limited	-	-	-	-	TIMEOUT	-
5	Uniform Cost	18234	18236	159707	12	62.17	YES
6	Recursive BFS	-	-	-	-	TIMEOUT	-
7	Greedy Best First	5606	5608	49369	22	19.18	NO
8	A*	18234	18236	149707	12	62.13	YES
9	A* ignore precond.	5040	5042	44944	12	19.81	YES
10	A* levelsum	-	-	-	-	TIMEOUT	-

Optimal path:
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(C2, P2, SFO)
Unload(C1, P1, JFK)

In problem 3, complexity rises again. Surprisingly, DFS was the fastest search, However, it is far from optimal regarding the length of the solution. BFS took a very long time but arrived at an optimal solution. Greedy best first search was among the faster implementations, but did not arrive at an optimal solution. In addition to previous timeouts, now A* levelsum also timed out at problem 3.

Comparison of non-heuristic searches

For non-heuristic searches we will compare BFS, DFS, and Depth Limited Search. BFS arrived always at an optimal solution, due to its nature of going through all possible solutions at certain depths first. Unfortunately, this makes it also a comparatively slow algorithm.
DSF on the other hand has good chances of finding a solution quickly. However, since it is looking deep down in the search trees first, it is seldom an optimal solution.

Depth Limited should in a lot of applications be the best of both worlds from BFS and DFS. Looking in for a DFS answer while being limited in depth, should find (close to) optimal solutions while keeping resources needed in check. However, with this particular problem, the search limited search appears not to function too well. It only finds a (non-optimal) solution for the first problem and times out at the other two. I suspect this is due to massive search space that allows BFS to come up with a good answer, DFS with a quick answer but Limited search goes comparatively deep without the benefits of going through all nodes.

Comparison of heuristic searches

Heuristic searches appear to be doing generally well on these. A* search finds optimal solutions at runtimes at or below BFS. When preconditions are ignored, solutions are found even quicker, but potentially at the cost of not being executable. A* with a levelsum heuristic is surprisingly resource hungry and therefore takes some time or can even time out. This could be an issue with the way the levelsum heuristic is implemented. Solutions are optimal as well.

Conclusion

Given the nature of this problem, different search types are more or less helpful depending on the complexity. On low levels of complexity, I would recommend going for A* searches. While processing times are higher than other searches, this approach finds an optimal solution. Since airplanes are sent to

different airports based on these findings, the cost for additional processing power is well compensated for over a non-optimal solution. Not accounting for preconditions might also cost more in the long run, even if the search itself tends to be quicker than a plain A*.

An informed search should generally be preferred over an uninformed search. Heuristics could be improved in the future, while uninformed searches like BFS would always need more resources with increasing complexity.