Planning Search Analysis

Different search methods were utilized to solve planning problems for air cargo transport systems with different initial states and goals. First, attempts were made to solve the problems without any heuristics, then with heuristics. The tables below describe the performance of the search methods for each of the problems. The optimal plan is the method that has the shortest plan length, followed by the fastest execution time and lowest memory usage.

Uninformed Search Analysis

Problem	Search Method	Expansions	Goal Tests	New Nodes	Plan length	Time (s)
1	Breadth-first	43	56	180	6	.032
	Depth-first	21	22	84	20	.018
	Greedy best first graph	7	9	28	6	.006
2	Breadth-first	3343	4609	30509	9	13.05
	Depth-first	624	625	5602	619	12.12
	Greedy best first graph	998	1000	8982	21	6.66
3	Breadth-first	14491	17947	128184	12	332.81
	Depth-first	2009	2100	17558	2014	23.36
	Greedy best first graph	4031	4033	35794	22	76.95

For the non-heuristic search methods, as shown above, breadth first search yields the most optimal plan as it has the shortest plan length for all three problems. Breadth first search will always be optimal in terms of plan length as it looks through all choices for every step of the plan. However, it also has the highest expansions, goal tests, new nodes, and execution time because it explores all these choices. According to section 3.4.1 of Artificial Intelligence: A Modern Approach 3e, Breadth search first is complete and optimal, but it is susceptible to high memory usage if the problem has a high branching factor, as shown by the performance of the algorithm in Problem 3.

In problem 1, greedy best first graph search is actually the most optimal as it has low execution time and yields the optimal plan length. However, in the other two problems, breadth-first search is the only method that achieves the optimal plan length. Depth first search is fastest for problem 3, whereas greedy best first graph search is fastest for problems 1 and 2.

Informed Search Analysis

Problem	Search Method	Expansions	Goal Tests	New Nodes	Plan length	Time (s)
1	A*h_1	55	57	224	6	.045
	A* h_ignore_preconditions	41	43	170	6	.04
	A*_h_pg_levelsum	11	13	50	6	4.6
2	A*h_1	4852	4855	44041	9	54.40
	A* h_ignore_preconditions	1506	1508	13820	9	14.31
	A*_h_pg_levelsum	86	88	841	9	1275.67
3	A*h_1	17783	17785	155920	12	548.60
	A* h_ignore_preconditions	5081	5083	45292	12	109.88
	A*_h_pg_levelsum	404	406	3718	12	7165.09

All three search methods generated optimal plan lengths. A* search with levelsum heuristic was most efficient in terms of memory usage, as it required the lowest number of expansions, goal tests, and new nodes. However, it also took the longest time to run by far, exceeding 10 minutes in both problems 2 and 3. Although it was efficient in memory usage, I would not consider it the most optimal merely due to its long execution time. Thus, the most optimal search method using heuristics would be A* search with the ignore preconditions heuristic, as it had a lower number of expansions, goal tests, and new nodes compared to A* search with h_1 heuristic. It also has the lowest execution time of all the informed search methods.

Conclusion

The results above show that an informed search using custom heuristics greatly outperforms uninformed search in terms of finding the optimal solution. If we compare breadth first search with A* search using the ignore_preconditions heuristic, the execution time is comparable in problems 1 and 2, with A* search significantly better for problem 3. A* search with the ignore_preconditions heuristic is also more efficient with lower number of expansions, goal tests, and new nodes compared to breadth first search. Thus, the most efficient search method overall would be A* search with ignore_preconditions heuristic.

Optimal Sequence of Actions Using A* Search with ignore preconditions heuristic

Problem 1:

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

Problem 2:
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)

Problem 3

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)