PLANNING SEARCH ANALYSIS

Optimal Plan for Problem 1 using Greedy Best First Graph

Using Greedy Best First Graph search would result in this plan:

Expansions	Goal Tests	New Nodes	Plan Length
7	9	28	6

This results in fewest expansions and fewest number of new nodes. For this problem, the greedy algorithm performed better than plain BFS .

Goal of problem 1 can be met as follows:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Optimal Plan for Problem 2 using Breadth First Search (BFS)

By using BFS , this plan is obtained:

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

Using greedy best first search resulted in a plan length of 21, and using DFS cost us as plan length of 105. Simple Breadth first search was the most optimal plan with plan length of 9. That said, BFs result did result in a lot of node expansions, almost 30 times the size of DFS and about 3 times the number of nodes expanded by greedy best first search.

Expansions	Goal Tests	New Nodes	Plan Length
3346	4612	30534	9

Optimal Plan for Problem 3 using breadth_first_search (BFS)

Here again, BFS resulted in a minimal plan length of 12. As before, DFS outperformed BFS in terms of nodes expansion, but the number of actions it took to reach the goal was significantly higher (292) compared to BFS tally of 12. Since the task here is to pick the most optimal plan, we go with BFS again.

Expansions	Goal Tests	New Nodes	Plan Length
14120	17673	124926	12

This is the plan with BFS:

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

Non-Heuristic Search Comparison

Planning	Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (seconds)
Problem 1	BreadFirst Search	44	57	184	6	0.042
	Depth First Search	21	22	84	20	0.016
	Depth Limited Search	96	248	391	50	0.104
Problem 2	BreadFirst Search	3346	4612	30534	9	16.05
	Depth First Search	107	108	959	105	0.415
	Depth Limited Search	-	-	-	-	-
Problem 3	BreadFirst Search	14120	17673	124926	12	121.5
	Depth First Search	292	293	2388	288	1.6
	Depth Limited Search	-	-	-	-	-

Observations

As we can see , the algorithm performance is really dependent on the problem at hand. When there are relatively few fluents (as in problem 1), Breadth-first search comes up with the most efficient plan in the shortest amount of time. As the planning space gets bigger, BFS takes longer to come up with the final plan, although it still produce the most optimal plan compared to other two algorithms.

Also, with bigger planning problem, large number of fluents and high number of state transitions, depth first search seems to outperform BFS handily in terms of processing latency. But in doing so, it performs lot more action and thus has a much longer plan length than BFS. On the other hand, BFS computes the most efficient plan length , but it comes at a huge cost of memory in storing the nodes.

Note: Depth limited Search for problem 2 and 3 ran indefinitely, with no results printed.

A* SEARCH COMPARISON

Planning	Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (seconds)
Problem 1	A* Search ignore_preconditions	41	43	170	6	0.048
	A* Search level-sum	11	13	50	6	0.98
	A* Search h1	55	57	224	6	0.079
Problem 2	A* Search ignore_preconditions	1450	1452	13303	9	5.18
	A* Search level-sum	86	88	841	9	84.9
	A* Search h1	4853	4855	44041	9	13.9
Problem 3	A* Search ignore_preconditions	5040	5042	44944	12	20.4
	A* Search level-sum	325	327	3002	12	397.5
	A* Search h1	18223	18225	159618	12	58.09

Observations

From the table above, we can see that *ignore-preconditions* heuristic is significantly faster than *level-sum heuristic,* although both return optimal plan. This is because in *ignore-preconditions* , every action at each state is considered valid and therefore, satisfying a goal is simply matter of checking what actions are not satisfied yet in the current state. That's a straightforward computation.Whereas, the *level-sum heuristic* involves takes longer because of multiple loops in its computation. As a result , the level-sum algorithmic complexity is higher. But the advantage it offers is that fewer nodes are visited and expanded.

Best Pick

Depth-First Search algorithm is not ideal for these kind of problem because it is not an optimal algorithm (as highlighted in Lesson 11, Lecture 22 on Search). The nature of DFS is that it may result in going down the action path that may result in the some of the goals not being met especially when the fluents in the planning space get large. Additionally, we assume every action in these planning graphs to have a unit cost. If the costs were uneven, the suboptimal solution of DFS would be all the more evident.
Hence, it can never be the best choice for this planning problem.

Looking at the output of the heuristic vs non-heuristic algorithms further, it can be seen that the two best performing algorithms in terms of optimal plan are : **A* search with level-sum** and **Breadth-First-Search(BFS).**

Both these algorithms produce identical optimal plan for problem 1,2 and 3 with plan lengths 6,9 and 12 respectively. However, we notice that level-sum achieves the same result with noticeably lower node expansions and new node creations than BFS. Although level-sum heuristic is inadmissible (Section 11.4, Chapter 11, AIMA) , it works well for this problem where subgoal independence holds true and the problem is decomposable as well
For producing the most optimal plan with fewest node expansions, **A* search with level-sum** is my choice.