Performance analysis

Testing result

Air Cargo Problem 1

Optimal solution:

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Unload(C1, P1, JFK)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)

Performance:

No.	Problem	Expansions	Goal Tests	New Nodes	Time Elapsed (s)	Plan length
1.	A* search with ignore preconditions heuristic	41	43	170	0.0950	6
2.	A* search with level-sum heuristic	11	13	50	0.5920	6
3.	Breadth first search	43	56	180 0	0.0697	6
4.	Depth first graph search	21	22	84	0.0398	20
5.	Depth limited search (limit depth = 50)	101	271	414	0.10555	50

Air Cargo Problem 2

Optimal solution:

1. Load(C3, P3, ATL)
2. Fly(P3, ATL, SFO)
3. Unload(C3, P3, SFO)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)
7. Load(C1, P1, SFO)
8. Fly(P1, SFO, JFK)
9. Unload(C1, P1, JFK)

Performance

No.	Problem	Expansions	Goal Tests	New Nodes	Time Elapsed (s)	Plan length
1.	A* search with ignore preconditions heuristic	1450	1452	13303	4.22	9
2.	A* search with level-sum heuristic	86	88	841	45.48	9
3.	Breadth_first_search	3343	4609	30509	13.21	9

No.	Problem	Expansions	Goal Tests	New Nodes	Time Elapsed (s)	Plan length
4.	Depth first graph search	624	625	5602	3.37	619
5.	Depth limited search (limit depth = 50)	222719	2053741	2054119	883.09	50

Air Cargo Problem 3:

Optimal solution:

1. Load(C2, P2, JFK)
2. Fly(P2, JFK, ORD)
3. Load(C4, P2, ORD)
4. Fly(P2, ORD, SFO)
5. Unload(C4, P2, SFO)
6. Load(C1, P1, SFO)
7. Fly(P1, SFO, ATL)
8. Load(C3, P1, ATL)
9. Fly(P1, ATL, JFK)
10. Unload(C3, P1, JFK)
11. Unload(C2, P2, SFO)
12. Unload(C1, P1, JFK)

Performance

No.	Problem	Expansions	Goal Tests	New Nodes	Time Elapsed (s)	Plan length
1.	A* search with ignore preconditions heuristic	5040	5042	44944	16.67	12
2.	A* search with level-sum heuristic	325	327	3002	230.83	12
3.	Breadth_first_search	14663	18098	129631	98.81	12
4.	Depth first graph search	408	409	3364	1.74	392
5.	Depth limited search (limit depth = 50)	---	---	---	> 1hr	---

Performance comparison

***Optimal solution*:**
The A* searches and breadth first search are able to find the optimal solution while the depth-first can't. In the Air cargo problem, all step costs are equal so the breadth first search always returns optimal solution if exists . The A* searches will find the optimal solution if the estimated cost function is admissible. The ignore-precondition heuristic is admissible because when ignoring the preconditions, all sub-goal can be reached with only 1 step. The level-sum heuristic, in general, is not admissible. But in the air cargo problems, the level-sum heuristic is admissible because all actions (fly ,load, unload) have only one positive effect. This means one action will never be able to reaches 2 or more sub-goals so the the level-sum always equal or less than the True cost. The depth-first searches explore the longest path first and return the first solution they founded so it's very unlikely that the founded solution is optimal.

Number of visited nodes:
The search with the highest number of visited nodes is ***Depth limited search (DLS)***. The ***DLS*** is a tree search so it does not keep track the visited nodes/states. This makes the ***DLS*** revisited some nodes many times. The ***A* search with ignore-precondition heuristic (ASIP)***, ***breath first search (BFS)*** and ***depth-first graph search (DFS)*** are graph searches so they keep track the visited nodes/states to avoid redundant node explorations. So these searches have much less visited nodes compare to ***DLS*** which is a tree search. The ***ASIP*** has the number of visited nodes lower than ***BFS***, cause its estimated distance heuristic seems do a good job in choosing which state is closer to the goal.
The ***A* search with level-sum heuristic (ASL)*** is carried out on the GRAPHPLAN data structure rather than the simple state graph used by ***ASIP, DFS and BFS***. In GRAPHPLAN scheme, the problems is organized into layers which are an state levels follow by an action level. This scheme combines the nodes/states and edges/actions in simple state graph into state_levels and action levels. This approach reduces the number of nodes and edges dramatically. The Mutex link in GRAPHPLAN encodes/represents the conflicts between Nodes/States and the conflict between edges/actions. The level-sum heuristic is also a good heuristic for A* search. All of these reasons explains why the ***ASL*** have the least visited nodes.

Elapsed time:
The elapsed time for each search method is closely related to the number of visited nodes. The more nodes to visit, the more time is required. An exception here is the case of ***ASL***. The ASL method had the least visited nodes but it takes more time to find the goal than DFS method. The reasons is that the construction of GRAPHPLAN data structure is computationally expensive.