The Problem

The project was to create a AI that can navigate through a statespace, from a initial state, to reach a goal state. Navigating through the states is done by search algorithms such as A* search, Breadth First Search, and depth first search just to name a few.

The problem that the Planning AI had to solve was getting Cargo to the right destination by plan. There were three problems to solve; each having a different number of planes, airports and cargoes. The Planner had 3 types of actions that it could do, Load, unload and fly.

The load Action had the effect of putting the cargo onto the plane and took 3 arguments; the cargo to be loaded, the airplane in which the cargo is meant to go and which airport both the cargo and plane were. The Unload Action which had the effect of taking a cargo off and putting it into the Airport took the same arguments as the Load action. The last action that the AI could do was Fly which had the effect of bringing one airport to another. This took 3 arguments, what plane was flying, Where it is flying from and where it is flying to. With all these actions the AI should be able to find a path to a goal state.

NON-heuristic findings

For the first problem (fig.1), the time, expansions, goal tests, and New Nodes results show that the depth first graph search was the best. This is good for systems that might be low on memory or processing power but would not give the most optimal path to the goal which was the breadth first search. This search function had half the path length of depth first graph search but had more then double of expanded nodes, goal tests, new nodes and time taken.

The Second problem had similar results to the first problem as looking at breadth first search and uniform cost search are more computationally heavy then depth first graph search. On the other had breadth first search and uniform search are far more efficient as instead of a path length of 1085, they have a short path length of nine.

Lastly problem 3 (fig.1), which had the most goals to achieve, it had the again similar results to the first two problems. Only difference is that the numbers are bigger. To conclude for non heuristic searches, the general trend of the problems run times and results increase as more sub-goals increase. Also breadth first search and uniform cost search result in the most optimal paths but are computationally heavy.

Heuristic based searches

both ignore preconditions and pg_levelsum give optimal path lengths as shown in figure 3 (the same lengths as breadth first search). The difference between the two are that ignore preconditions goes through more expansions, goal tests and new nodes by a significant amount then pg_levelsum; but ignore preconditions takes up a lot less time to get to the same result. This could be due to how each functions works as ignore preconditions your able to do moves in any order and could cause it to go through move moves as it doesn’t need to worry about weather or not a move preconditions are True. For pg_levelsum the time could be longer for the opposite reason then preconditions as level sum needs to check if the preconditions are true. The rest of the results could be lower, in level sum, because it just needs to see what level the goal is on then it goes to the next goal and sees where that is in the graph.

Conclusion

To conclude, the non heuristic searches were better for smaller problems such as problem one with only two goal states. The best search algorithm to use for smaller problems would be depth first graph searches as it gives better results by a significant margin in all factors. For more complex problems Heuristic based searches would be more suitable as they can get better results in all factors. The best search algorithm for larger problems would be the pg_levelsum if time is not critical. If time is critical then ignore preconditions would be better tow use.

Results

Fig. 1

Problem 1	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
breadth_first_search	43	56	180	0.028	6
Depth First Graph search	12	13	48	0.007	12
uniform_cost_search	55	57	224	0.036	6
///					
Problem 2	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
breadth_first_search	3,346	4,612	30,534	14.672	9
Depth First Graph search	1,124	1,125	10,017	7.902	1085
uniform_cost_search	4,853	4,855	44,041	12.516	9
///					
Problem 3	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
breadth_first_search	14,663	18,089	129,631	102.905	12
Depth First Graph search	627	628	5,176	3.280	596
uniform_cost_search	18,235	18,237	159,716	57.404	12

Fig. 2

Problems	**breadth_first_search Paths**
Problem 1	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)
Problem 2	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)
Problem 3	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)

fig. 3

Problem 1	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
A* h_ignore_preconditions	41	43	170	0.059	6
A* H_Pg_levelsum	11	13	50	1.293	6
//					
Problem 2	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
A* h_ignore_preconditions	1450	1452	13303	7.639	9
A* h_pg_levelsum	86	88	841	316.678	9
//					
Problem 3	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path length**
A* h_ignore_preconditions	5040	5042	44944	25.572	12
A* h_pg_levelsum	325	327	3002	1694.808	12