Udacity - Artificial Intelligence Nanodegree - nd889

Problem 1

The first problem has 2 pieces of cargo, 2 airports and 2 planes. Each airport has a single piece of cargo and a plane, and the goal state is that the cargo is swapped between the airports.

Start

- SFO has C1 and P1
- JFK has C2 and P2

Goal

- JFK has C1
- SFO has C2

Solution

The most optimal solution to the problem is in 6 steps:

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

Algorithm Comparison

For this problem Greedy Best First Graph Search (GBFGS + h1) performs the best by far, achieving the fastest speed with the lowest number of node expansions. The reasoning for this is, "Greedy best-first search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is, f(n) = h(n)." **[1]** In this small problem GBF finds the solution easily with minimal node expansion or time.

Algorithm	Nodes Exp	Goal Tests	New Nodes	Steps	Time (s)
BFS	43	56	180	6	0.068
BFTS	1458	1459	5960	6	2.044
DFGS	21	22	84	20	0.030
DLS	101	271	414	50	0.160
UCS	55	57	224	6	0.077
RBFS + h1	4229	4230	17023	6	5.777
GBFGS + h1	**7**	**9**	**28**	**6**	**0.009**
A* + h1	55	57	224	6	0.082
A* + hIgnore	41	43	170	6	0.062
A* + hLevelsum	11	13	50	6	6.622

Problem 2

The second problem has 3 pieces of cargo, 3 airports and 3 planes. Each airport has a single piece of cargo and a plane, and the goal state is that the cargo is moved.

Start

- SFO has C1 and P1
- JFK has C2 and P2
- ATL has C3 and P3

Goal

- JFK has C1
- SFO has C2 and C3

Solution

The most optimal solution to the problem is in 9 steps:

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

Algorithm Comparison

For this problem even though GBFGS is still the fastest it doesn't find the most optimal solution. BFS is able to solve an optimal solution in only 6.5 times longer, because "if the shallowest goal node is at some finite depth d, breadth-first search will eventually find it after generating all shallower nodes (provided the branching factor b is finite)" **[2]**

However we can already see that the time and space complexity is becoming a problem for BFS. Turning to A* with the ignore heuristic we can solve the problem with an optimal solution and only 4.5 times the GBFGS run time and less than half the node expansion of BFS. Even though the problem appears to be only marginally more complex than the first, non optimal solutions can be drastically worse for cost, so if optimal step count is important certain algorithms such as DFGS should be ruled out.

BFTS, DLS, RBFS and A* + hLevelsum did not complete in under 10 minutes.

Algorithm	Nodes Exp	Goal Tests	New Nodes	Steps	Time (s)
BFS	3343	4609	30509	9	24.647
BFTS	-	-	-	-	Timeout
DFGS	624	625	5602	619	5.406
DLS	-	-	-	-	Timeout
UCS	4604	4606	41828	9	59.891
RBFS + h1	-	-	-	-	Timeout
GBFGS + h1	455	457	4095	16	3.763

A* + h1	4604	4606	41828	9	60.582
A* + hIgnore	**1398**	**1400**	**12806**	**9**	**16.995**
A* + hLevelsum	-	-	-	-	Timeout

Problem 3

The third problem has 4 pieces of cargo, 4 airports and 2 planes. Each airport has a single piece of cargo and two have planes, and the goal state is that all cargo is moved to two airports.

Start

- SFO has C1 and P1
- JFK has C2 and P2
- ATL has C3
- ORD has C4

Goal

- JFK has C1 and C3
- SFO has C2 and C4

Solution

The most optimal solution to the problem is in 12 steps:

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, ATL)
3. Load(C3, P1, ATL)

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

Algorithm Comparison

Finally we can see here achieving an optimal result in a reasonable amount of time is best served by A* and a fast heuristic. The ignore preconditions heuristic "drops all preconditions from actions. Every action becomes applicable in every state, and any single goal fluent can be achieved in one step" **[3]**

This provides a very quick estimate of the how close any given state is to the goal state, ignoring preconditions.

BFTS, DLS, RBFS and A* + hLevelsum did not complete in under 10 minutes.

Algorithm	Nodes Exp	Goal Tests	New Nodes	Steps	Time (s)
BFS	14663	18098	129631	12	186.433
BFTS	-	-	-	-	Timeout
DFGS	408	409	3364	392	3.494
DLS	-	-	-	-	Timeout
UCS	16963	16965	149136	12	519.007

RBFS + h1	-	-	-	-	Timeout
GBFGS + h1	3998	4000	35002	30	95.916
A* + h1	16963	16965	149136	12	520.158
A* + hIgnore	**4723**	**4725**	**41835**	**12**	**100.605**
A* + hLevelsum	-	-	-	-	Timeout

References

[1] Chapter 3.5 Informed (heuristic) Search Strategies - Page 92. AIMA 3rd Edition.

[2] Chapter 3.4 Uninformed Search Strategies - Page 81. AIMA 3rd Edition.

[3] Chapter 10.2 Algorithms For Planning As State-space Search - Page 376. AIMA 3rd Edition.