Heuristic Analysis

Air Cargo Transport

Udacity Artificial Intelligence Nanodegree

By: Mayank Prakash

Artificial Intelligence Nanodegree

Air Cargo Transport - Heuristics Analysis

Problem 1:

The first problem requires planning of 2 cargos to be transported to 2 airport via 2 airplane.

Start:

Cargo C1, Plane P1 is at SFO

Cargo C2, Plane P2 is at JFK

Goal:

C1 is at JFK

C2 is at SFO

Solution:

The minimum number of steps to be followed to obtain a solution is as follows:

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

Comparison:

Algorithm	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time
BFS	43	56	180	6	0.0494
BFTS	1458	1459	5960	6	1.4953
DFGS	21	22	84	20	0.0364
DLS	101	271	414	50	0.1460
UCS	55	57	224	6	0.0604
RBFS	4229	4230	17023	6	4.7811
GBFGS	**7**	**9**	**28**	**6**	**0.0085**
A*	55	57	224	6	0.0567
A* ignore precondition	41	43	170	6	0.0452
A* level_sum	11	13	50	6	1.0431

Conclusion:

For this problem **Greedy Best First Graph Search** outperforms all other algorithms as it tries to achieve the goal by expanding nodes that are closest to the goal in order to achieve the solution faster.

Problem 2:

This problem is like first problem with an additional complexity. In this case we have 3 cargos to be transported to 2 airports via 3 planes.

Start:

Cargo C1, Plane P1 is at SFO

Cargo C2, Plane P2 is at JFK

Cargo C3, Plane P3 is at ATL

Goal:

C1 is at JFK

C2 and C3 are at SFO

Solution:

The minimum number of steps to be followed to obtain a solution is as follows:

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

Comparison:

Algorithm	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time
BFS	3343	4609	30509	9	13.6222
BFTS	—	—	—	—	Timeout
DFGS	624	625	5602	619	5.8512
DLS	222719	2053741	2054119	50	2306.1394
UCS	4604	4606	41828	9	30.1922
RBFS	—	—	—	—	Timeout
GBFGS	455	457	4095	16	4.1239
A*	4604	4606	41828	9	41.3827
A* ignore precondition	1310	1312	11979	9	12.3739
A* level_sum	**74**	**76**	**720**	**9**	**384.6938**

Conclusion:

For this problem **A* level_sum** outperforms all other algorithms, although the time complexity to achieve the goal state is quite high. Like previous case, although Greedy Best First Graph Search performs better than others, except A* level_sum, we cannot rely on it to provide us with the optimal solution as we can see the path length is clearly more than what is expected to be minimum. We should also take into consideration the performance of Depth First Greedy Search. The depth wise greedy approach gives this algorithm a way to perform faster despite very long path length. So despite better performance we won't take this into consideration.

Algorithms such as A*, Uniform Cost Search and Breadth First Search suffer from issue of large space complexity which can be observed from total new nodes that are formed.

Algorithms such as Breadth First Tree Search and Recursive Best First Search fail to achieve the goal state before timeout. Although Depth Limited Search returns but time taken is too much which isn't feasible.

Problem 3:

This problem is like first and second problem with additional complexities. In this case we have 4 cargos to be transported to 2 airports via 2 planes.

Start:

Cargo C1, Plane P1 is at SFO

Cargo C2, Plane P2 is at JFK

Cargo C3 is at ATL

Cargo C4 is at ORD

Goal:

C1 and C3 are at JFK

C2 and C4 are at SFO

Solution:

The minimum number of steps to be followed to obtain a solution is as follows:

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

Comparison:

Algorithm	Nodes Expanded	Goal Tests	New Nodes	Plan Length	Time
BFS	14663	18098	129631	12	158.1081
BFTS	—	—	—	—	Timeout
DFGS	408	409	3364	392	6.3868
DLS	—	—	—	—	Timeout
UCS	16963	16965	149136	12	210.9201
RBFS	—	—	—	—	Timeout
GBFGS	3998	4000	35002	30	61.6375
A*	16963	16965	149136	12	254.2476
A* ignore precondition	**4444**	**4446**	**39227**	**12**	**67.1878**
A* level_sum	229	231	2081	13	2689.5244

Conclusion:

For this problem, since the problem has greater complexity when compared to the previous 2, there are several scenarios to be taken into consideration.

- **DFGS**: This algorithm traverses the nodes very fast, the time complexity is very decent but because of the path length, this doesn't add up as a viable choice.
- **GBFGS:** This can serve as one of the option, although it doesn't provide an optimal path, but it doesn't create a lot of new nodes as done by DFGS, and does terminate with a solution in reasonable amount of time.
- **A* ignore precondition:** This heuristic returns an optimal path length, the number of nodes explored is higher than number of nodes explored in GBFGS, but it doesn't vary by a very large amount so this can be considered over GBFGS given a choice.

A* suffers from space complexity issue in this particular problem and so does Uniform Cost Search, although the time complexity is acceptable but space is still an issue.

In case of uninformed strategies, since the search doesn't have memories except of those provided in the problem, and its ability to distinguish, whether state is a goal state or not, for **Uniform Cost Search** the path with lowest path cost is expanded and goal test is applied to the entire expansion, which requires a space to store the expanded nodes, which adds an overhead to the space. The fact is UCS doesn't care about the space or anything for that matter, it just tries to minimise the cost.

For **A* search**, like UCS, along with the path cost it also takes into consideration the estimated cost of the cheapest path. This results in larger expansion of nodes, in order to search for and minimise cheapest path and path cost, there by increasing the space complexity despite its acceptable time limit.

A* level_sum despite the best performance in terms of number of nodes expanded and nodes to reach goal and new nodes formed, it is space efficient but not at all a feasible choice because of the time taken to terminate and reach goal state.