Heuristic Analysis for the Planning Search Project

The project asks us to setup the problems for search, experiment with various automatically generated heuristics, including planning graph heuristics, to solve the problems, and then provide an analysis of the results.

Problem 1: Two Airports, Planes and Cargos

Start:
Cargo C1 is at SFO
Cargo C2 is at JFK
Plane P1 is at SFO
Plane P2 is at JFK

Goal:
Cargo C1 is at JFK
Cargo C2 is at SFO

Solution:

Optimal Solution has 6 steps-

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

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time
BFS	43	56	180	6	0.03861
DFGS	12	13	48	12	0.01157
UCS	55	57	224	6	0.04075
GBFGS w/ h_1	7	9	28	6	0.00456
A* w/ h_1	55	57	224	6	0.04283
A* w/ h_ignore_pre	41	43	170	6	0.03162
A* w/ h_pg_levelsum	11	13	50	6	0.71610

Analysis:

GBFGS w/ h_1 works best for this problem in terms of minimum number of expansions, goal tests, new nodes. Execution time is also the best since this algorithm is goal-directed and it uses just the heuristic function. It works for the small problem. BFS is also optimal, but expands more nodes as it is not goal-directed.

Problem 2: Three Airports, Planes and Cargos

Start:

Cargo C1 and Plane P1 are at SFO
Cargo C2 and Plane P2 are at JFK
Cargo C3 and Plane P3 are at ATL

Goal:

Cargo C1 is at JFK
Cargo C2 is at SFO
Cargo C3 is at SFO

Solution:

Optimal Solution has 9 steps-

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

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

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time
BFS	3343	4609	30509	9	24.7572
DFGS	624	625	5602	619	5.0936
GBFGS w/ h_1	455	457	4095	16	2.0034
A* w/ h_ignore_pre	1310	1312	11979	9	9.2636
A* w/ h_pg_levelsum	74	76	720	9	80.0000

Analysis:

GBFGS w/ h_1 is the fastest algorithm for this problem, but it does not find the optimal solution - plan length of 16 v/s 9. BFS finds the optimal solution, but time and space complexity is an issue because this problem is more complex than Problem 1.

A* w/ h_ignore_pre is the overall best algorithm in terms of minimum number of expansions, goal tests, new nodes.

Problem 3: Four Airports, Two Planes and Four Cargos

Start:

Cargo C1 and Plane P1 are at SFO
Cargo C2 and Plane P2 are at JFK
Cargo C3 and Plane P3 are at ATL

Goal:

Cargo C1 is at JFK
Cargo C2 is at SFO
Cargo C3 is at SFO

Solution:

Optimal Solution has 12 steps

Load(C1, P1, SFO)
Load(C2, P2, JFK)
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)

Algorithm	Expansions	Goal Tests	New Nodes	Plan length	Time
BFS	14663	18098	129631	12	148.5882
DFGS	408	409	3364	392	2.8734
GBFGS w/ h_1	3998	4000	35002	30	21.8050
A* w/ h_ignore_pre	4444	4446	39227	12	24.5878
A* w/ h_pg_levelsum	229	231	2081	12	351.9832

Analysis:

The results clearly indicate that using informed search strategies with custom heuristics perform better overall especially for bigger complex problems.

A* w/ h_ignore_pre is the best algorithm in terms of minimum number of expansions, goal tests, new nodes and execution time. It drops all preconditions from actions. Every action can be executed in every state and any goal can be achieved in one step. This gives a very quick estimate of how close a given state is to the goal state. It is optimal.

The h_pg_levelsum heuristic returns the sum of the level costs of the goals. It is much more accurate than the ignore preconditions heuristic. It is significantly slower for this problem because it requires the planning graph to be generated.

DFS is efficient in memory usage and is one of the quickest. But it doesn't necessarily find the optimal path because it visits deeper nodes first (LIFO) and takes a meandering path to the goal which might be closer to the start. In fact, it fails in infinite-depth spaces.

BFS is guaranteed to find an optimal solution as it goes level by level, but memory usage is high since it expands many new nodes.

References:

[1] Artificial Intelligence: A modern approach Third Edition. Norvig and Russel