Heuristic Analysis for the Project -3 AIND
Implementing a Planning Search

By: Abuhanif Bhuiyan, Feb 2018

Introduction:

This project includes the application of AI in planning and search algorithm by comparing different search algorithm performance for three given problems. Also, to evaluate the effect of different heuristic function on the optimal solutions. Description of the function applications their related performances are explained below.

Results and Analysis:

I found the following optimal solutions for the problems. Problem-1 required 6 steps, problem-2 required 9 steps and problem-3 required 12 steps shown below.

Problem-1 (6 Steps)	**Problem-2 (9 Steps)**	**Problem-3 (12 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) Elapsed time: 0.017 sec	Load (C2, P2, JFK) Load (C1, P1, SFO) 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) Elapsed time: 12.04sec	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) Elapsed time: 49.9sec

Table-1: Optimal solutions for the problems.

As picture tells a thousand words. To better understand the scenario, I drew the following figure of the initial state to final goal of the cargos.

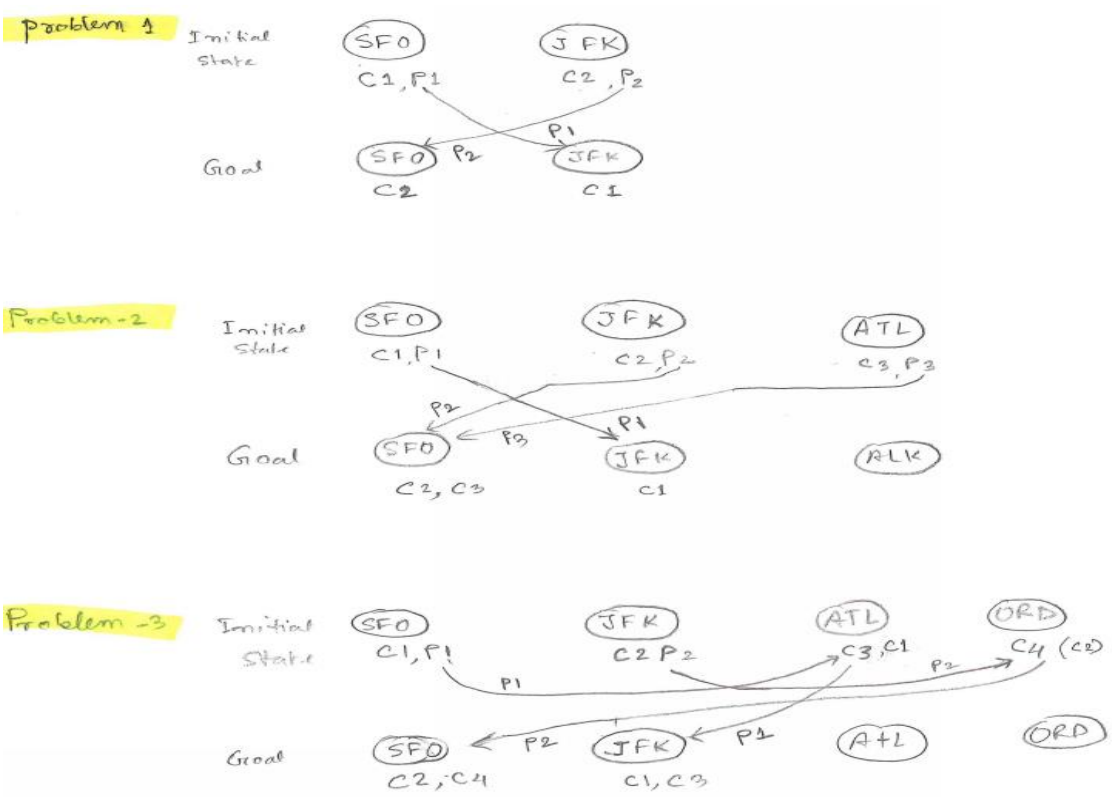

Fig1: Planning search graphical representation of the initial state to final goal.

Now I like to compare the results based on the algorithm and the heuristic functions.

Search	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
Breadth_first_search	43	56	180	6	0.1
Breadth_first_tree_search	1458	1459	5960	6	2.62
Depth_first_graph_search	12	13	48	12	0.018
Depth_limited_search	101	271	414	50	0.2
Uniform_cost_search	55	57	224	6	0.08
Recursive_best_first_search	4229	4230	17029	6	7.12
Greedy_best_first_graph_search	7	9	28	6	0.017
A* using h1 heuristic	55	57	224	6	0.12
A* using h_ignore_preconditions	41	43	170	6	0.13
A* using h_pg_levelsum	11	13	50	6	1.9

Table-2 Results from the problem-1

As expected Problem1 was the simplest and execution time were faster compared to the other two problems. The non-heuristic search algorithms were quick, with optimal solution came from several algorithms, but not all. GBFGS showed the best time optimum solution. In contrast, every heuristic search algorithm found an optimal plan. A* using h_pg_levelsum, was able to get the optimum solution with less new nodes. The fact that A* Search h_ignore_preconditions was able to find the optimal plan and beat BFS in terms of expansions, goal tests, new nodes, and time elapsed shows the power of even a simple heuristic such as calculating the number of remaining goals.

Search	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
Breadth_first_search	3343	4609	30509	9	15.21
Depth_first_graph_search	582	583	5211	575	9.3
Greedy_best_first_graph_search	990	992	8910	15	6.9
A* using h1 heuristic	4852	4854	44030	9	33.9
A* using h_ignore_preconditions	1450	1452	13303	9	12.04
A* using h_pg_levelsum	86	88	841	9	130.96

Table-3 Results from the problem-2

For Problem 2, the heuristic-search algorithms performed better with the A* Search h_ignore_preconditions than all its peers in terms of time elapsed, and A* Search h_pg_levelsum outperforming in terms of expansions, goal tests, and new nodes. This is because the planning graph grew in terms of cargo, planes and airports, making simple non-heuristic searches less efficient since they have to explore every node.

Search	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time**
Breadth_first_search	14663	18098	129631	12	117.7
Depth_first_graph_search	627	628	5176	596	8.9
Greedy_best_first_graph_search	5614	5616	49429	22	44.38
A* using h1 heuristic	18235	18237	159716	12	140.2
A* using h_ignore_preconditions	5040	5042	44944	12	49.9
A* using h_pg_levelsum	316	318		12	367.6

Table-4 Results from the problem-3

For Problem 3, A* Search h_ignore_preconditions outperformed all its peers since its main competitors all timed out due to the increased number of states in both the initial and goal state. It shows the exponential run-time of planning problems just by adding one more initial and goal state would render many of the algorithms too slow. Furthermore, simpler algorithms like BFS still have a place given that they scale much better.

Conclusion:

Among the non-heuristic search functions, breadth first search and uniform cost search algorithms are the most optimal. Though Depth first graph search is faster, it doesn't give an optimal solution. Among the heuristic solutions we can say that most of the times the **A* using h_ignore_preconditions** will be the best choice, it gives us the optimal solution with the best execution time and better memory usage in comparison to optimal non-heuristics searches. If we must use less memory then **A* using h_pg_levelsum** is our best choice because it is still optimal and have the best usage of the memory, with longer execution time