AIND Project 3: Planning Solution Search Analysis

Arun K Viswanathan

3/12/2017

Overview

This paper discusses the solutions for the three planning problems in the Air Cargo domain described in README.md.

Problem 1

The optimal sequence of actions to solve this problem has 6 steps which are show below:

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

This plan was discovered using the following searches and heuristics: breadth_first_search, breadth_first_tree_search, uniform_cost_search, recursive_best_first_search with h_1, greedy_best_first_graph_search with h_1, astar_search with h_1, astar_search with h_ignore_preconditions, astar_search with h_pg_levelsum.

The table below shows how various searches and heuristics performed on this problem:

Heuristic	Expansions	Goal.Tests	New.Nodes	Plan.Length	Time
breadth_first_search	43	56	180	6	0.030
breadth_first_tree_search	1458	1459	5960	6	1.082
depth_first_graph_search	21	22	84	20	0.015
depth_limited_search	101	271	414	50	0.100
uniform_cost_search	55	57	224	6	0.041
recursive_best_first_search with h_1	4229	4230	17023	6	2.907
greedy_best_first_graph_search with h_1	7	9	28	6	0.005
astar_search with h_1	55	57	224	6	0.057
astar_search with h_ignore_preconditions	41	43	170	6	0.077
astar_search with h_pg_levelsum	11	13	50	6	2.649

For this problem, the greedy best first search is fastest and also finds the same optimal solution discovered by breadth first search. Given the very small number of fluents, the greedy algorithm has very little work in reaching the solution.The depth first graph search and depth limited search produce plans with a large number of steps since they search redundant actions (e.g: Load followed by an immediate Unload) which effectively don't change the state.

Problem 2

The optimal sequence of actions to solve this problem has 9 steps which are show below:

```
Load(C1, P1, SFO)
```

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

This plan was discovered using the following searches and heuristics: breadth_first_search, uniform_cost_search, astar_search with h_1, astar_search with h_ignore_preconditions, astar_search with h_pg_levelsum.

The table below shows how various searches and heuristics performed on this problem:

	Heuristic	Expansions	Goal.Tests	New.Nodes	Plan.Length	Time
11	breadth_first_search	3343	4609	30509	9	14.340
13	depth_first_graph_search	624	625	5602	619	3.367
15	uniform_cost_search	4853	4855	44041	9	43.716
17	greedy_best_first_graph_search with h_1	998	1000	8982	21	7.774
18	astar_search with h_1	4853	4855	44041	9	50.688
19	astar_search with h_ignore_preconditions	1506	1508	13820	9	28.742
20	astar_search with h_pg_levelsum	86	88	841	9	260.208

For this problem, three searches do not run in a reasonable amount of time - breadth first tree search, depth limited search and recursive best first search with the h1 heuristic. In all these searches, redundant pairs of actions that don't chnage the state add to the size of the search space. Among the A* searches, the "ignore preconditions" heuristic produces the solution with the least time. The "levelsum" heuristic greatly reduces the number of expansions, goal tests and new nodes, but takes more time than the other two A* searches since the implementation of levelsum is very inefficient - it keeps recreating the plan graph every time it is invoked. Put another way, the big-O performance of the "levelsum" heuristic is best but the constants overwhelm the real performance of the implementation.

Problem 3

The optimal sequence of actions to solve this problem has 12 steps which are show below:

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

This plan was discovered using the following searches and heuristics: breadth_first_search, uniform_cost_search, astar_search with h_1, astar_search with h_ignore_preconditions, astar_search with h_pg_levelsum.

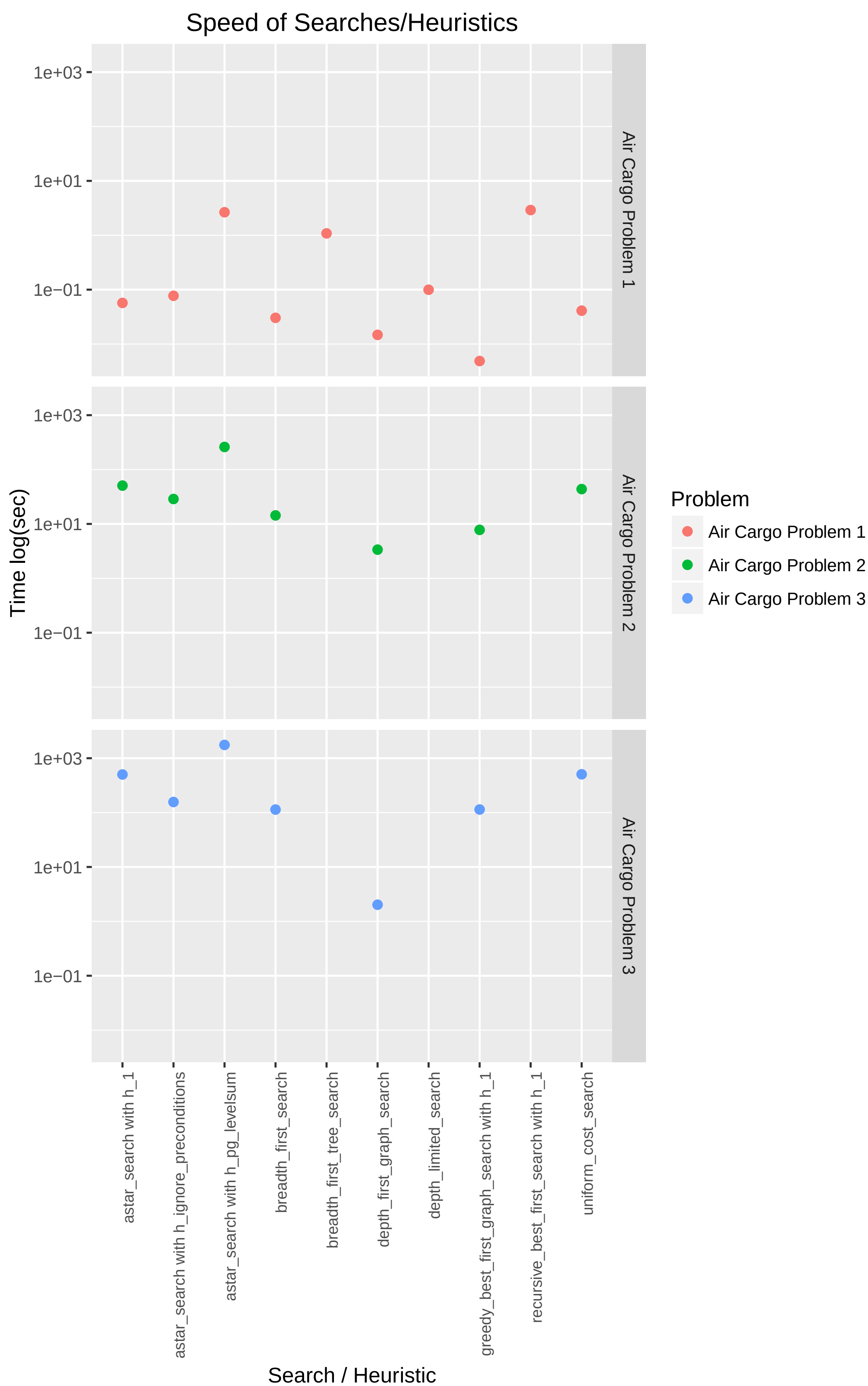
The table below shows how various searches and heuristics performed on this problem:

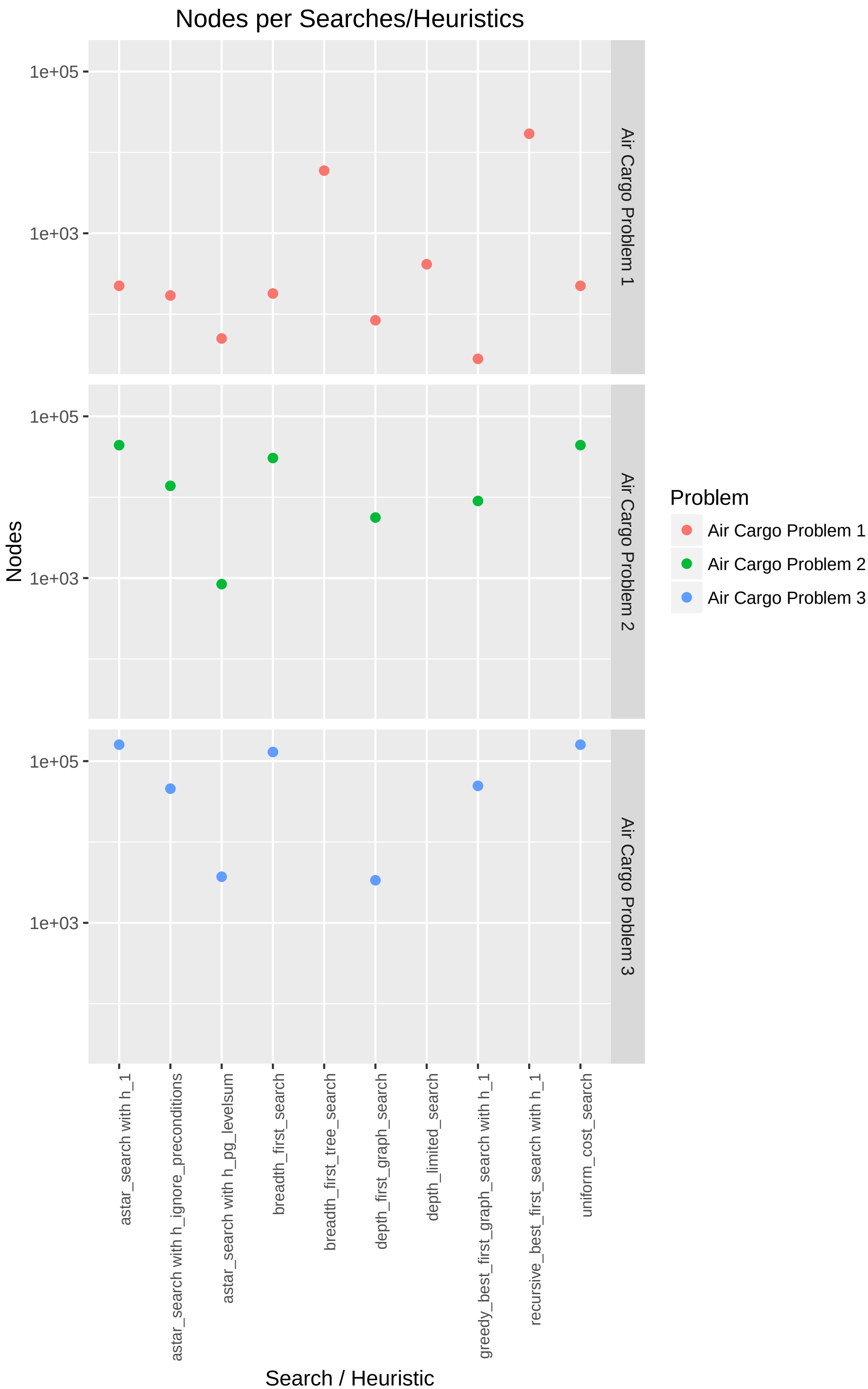
	Heuristic	Expansions	Goal.Tests	New.Nodes	Plan.Length	Time
21	breadth_first_search	14663	18098	129631	12	113.744
23	depth_first_graph_search	408	409	3364	392	2.022
25	uniform_cost_search	18235	18237	159716	12	506.124
27	greedy_best_first_graph_search with h_1	5614	5616	49429	22	113.938
28	astar_search with h_1	18235	18237	159716	12	502.283
29	astar_search with h_ignore_preconditions	5118	5120	45650	12	156.656
30	astar_search with h_pg_levelsum	404	406	3718	12	1755.833

Most of the observations for this problem are similar to problem 2. Again, three searches do not run in a reasonable amount of time - breadth first tree search, depth limited search and recursive best first search with the h1 heuristic. In all these searches, redundant pairs of actions that don't chnage the state add to the size of the search space. Among the A* searches, the "ignore preconditions" heuristic produces the solution with the least time. The "levelsum" heuristic greatly reduces the number of expansions, goal tests and new nodes, but takes more time than the other two A* searches since the implementation of levelsum is very inefficient - it keeps recreating the plan graph every time it is invoked. Put another way, the big-O performance of the "levelsum" heuristic is best but the constants overwhelm the real performance of the implementation.

Visualizations

The following pages show two plots: 1. a plot with the time taken by each of the searches/heuristics for the three problems. The y-axis uses a log scale to accentuate differences between the problems and searches. 2. a plot with the new nodes for the searches/heuristics for the three problems. The y-axis uses a log scale to accentuate differences between the problems and searches.


Speed of Searches/Heuristics
Time log(sec)
1e+03
1e+01
1e−01
Air Cargo Problem 1
Air Cargo Problem 2
Air Cargo Problem 3
Problem
astar_search with h_1
astar_search with h_ignore_preconditions
astar_search with h_pg_levelsum
breadth_first_search
breadth_first_tree_search
depth_first_graph_search
depth_limited_search
greedy_best_first_graph_search with h_1
recursive_best_first_search with h_1
uniform_cost_search
Search / Heuristic


Nodes per Searches/Heuristics
Nodes
1e+05
1e+03
Air Cargo Problem 1
Air Cargo Problem 2
Air Cargo Problem 3
Problem
Air Cargo Problem 1
Air Cargo Problem 2
Air Cargo Problem 3
astar_search with h_1
astar_search with h_ignore_preconditions
astar_search with h_pg_levelsum
breadth_first_search
breadth_first_tree_search
depth_first_graph_search
depth_limited_search
greedy_best_first_graph_search with h_1
recursive_best_first_search with h_1
uniform_cost_search
Search / Heuristic