Planning Heuristic Analysis

Andre Marinho

Let's compare different heuristics against our air cargo problem.

The Problem

We have to plan the transportation of cargo between airports. In summary, we have: cargos, airports and planes, and destinations for the cargos.

For comparing the heuristics we're going to run three simulations (problems):

Problems	cargos	planes	airports	goals	preconditions
Problem 1	2	2	2	2	4
Problem 2	3	3	3	3	6
Problem 3	4	2	4	4	6

Uninformed Search Heuristics

Let's first compare uninformed search heuristics. The selected heuristics where:

- Breadth First
- Deep First Graph
- Uniform Cost Search

Breath first tree search and deep limited search were discarded due to poor performance under the implementation of the planner algorithms. That would require a profiling analysis on the code.

Uninformed Search Results

Problem 1

	breadth_first_search	depth_first_graph_search	uniform_cost_search
Time elapsed (s)	0.1109612394	0.02880546798	0.1320952056
Plan length	6	12	6
Expansions	43	12	55
Goal Tests	56	13	57
New Nodes	180	48	224

Command execution: python run_search.py -p 1 -s 1 3 5

Problem 2

	breadth_first_search	depth_first_graph_search	uniform_cost_search
Time elapsed (s)	29.3555915	9.969280438	41.30760018
Plan length	9	575	9
Expansions	3343	582	4852
Goal Tests	4609	583	4854
New Nodes	30509	5211	44030

Command execution: python run_search.py -p 2 -s 1 3 5

Problem 3

	breadth_first_search	depth_first_graph_search	uniform_cost_search
Time elapsed (s)	141.5135189	10.4754847	194.1361332
Plan length	12	596	12
Expansions	14663	627	18223
Goal Tests	18098	628	18225
New Nodes	129631	5176	159618

Command execution: python run_search.py -p 3 -s 1 3 5

Breadth First and Uniform Cost has shown close results, with breadth first performing slightly better, due to around 20% less expansions which also means less goal tests.

The best performance in terms of time was the ‘deep first graph search’, it has a significantly higher number of plans, comparing to the other two heuristics, but much lower expansions, with less goal tests, and new nodes.

Uninformed Search Comparison

	Breadth First	Depth First	Uniform Cost
Time	B^D	B^M	B^D
Space	B^D	B^M	B^D
Optimal?	Yes	No	No

Complete?	Yes	No	Yes

Where: B = Branching factor; D = Depth of solution; M = Maximum depth of the search tree

Looking at the above comparison table, we are able to tell why breadth-first and uniform-cost has similar results, while depth first performs better in terms of time (and memory), but on other hand, it is neither optimal nor complete.

Informed Search Results

The selected heuristics where:

- A* search
- A* search with ignore preconditions
- A* search with level sum

Problem 1

	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Time elapsed(s)	0.1781511549	0.1784536291	10.00737659
Plan length	6	6	6
Expansions	55	41	11
Goal Tests	57	43	13
New Nodes	224	170	50

Command execution: python run_search.py -p 1 -s 8 9 10

Problem 2

	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Time elapsed (s)	49.99804999	18.59869426	5137.561399
Plan length	9	9	9
Expansions	4852	1450	86
Goal Tests	4854	1452	88
New Nodes	44030	13303	841

Command execution: python run_search.py -p 2 -s 8 9 10

Problem 3

	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Time elapsed (s)	216.3593256	71.62583328	-
Plan length	12	12	-
Expansions	18223	5040	-
Goal Tests	18225	5042	-
New Nodes	159618	44944	-

Command execution: python run_search.py -p 3 -s 8 9 10

On problem number 3, we weren't able to run 'A* with level sum', since it was still running after a couple of hours.

First, We've tried to improve the h_level_sum() function, but them realized that the issue was not there. Other parts of the code were also changed, but without proper analysis of where the bottleneck is.

The python optimizer (JIT) wasn't tested either. All the tests (informed and uninformed) where done in a Intel Pentium N3540, 2.16GHZ, with 4GB of memory, and spinning hard disk. A entry level Lenovo laptop.

We can notice that A* with ignore pre-conditions performs better than A*, as expected, but also fewer expansions, which leads to fewer goal tests and fewer new nodes. A trend we already noticed on the uninformed search, and pretty logical: fewer expansions leads to fewer new nodes and fewer goal tests.

But, unlike the uninformed search, it has not necessary being translated to better performance, since A* with levem sum had much fewer node expansions, but a significatively wort performance, for instance 85min against 1min.

As said earlier, a preliminary analysis indicated that the problem is not the code that does the level summing, which opens room for things like memory management.

Optimal Plans

After running all heuristics, these were the optimal sequence of actions for each problem.

Problem 1: 6 Steps

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

Problem 2: 9 Steps

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

Problem 3: 12 Steps

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

Informed vs Uninformed Search

Although Deep First Search, had the best time performance, it is neither complete nor optimal. For that reason, A star would be our choice, since it performed better than the other uniformed searches (Breadth first and uniform cost), and it is both complete and optimal.