Problem	# Positive fluents	# Negative fluents	Total fluents	State space size
1	4	8	12	2^{12}
2	6	21	27	2^{27}
3	6	26	32	2^{32}

Table 1: Number of fluents and state space size for each problems

Introduction

The aim of this report is to compare the performance of 10 heuristics for solving the air cargo problem. There are 3 problems with increased complexity (See table 1). The state space is given by $2^{\#fluents}$ since there 2 states for every fluent. So it is easy to see that by increasing only a little the number of fluents the state space gets very big.

Solutions for each problem

Since those problems are not difficult it is possible to get the best solution by hand.

Solution Problem 1

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Unload(C1, P1, JFK)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)

Total steps: 6

Solution Problem 2

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Unload(C1, P1, JFK)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)
7. Load(C3, P3, ATL)
8. Fly(P3, ATL, SFO)
9. Unload(C3, P3, SFO)

Total steps: 9

Solution Problem 3

1. Load(C1, P1, SFO)
2. Fly(P1, SFO, JFK)
3. Unload(C1, P1, JFK)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)
7. Load(C3, P3, ATL)
8. Fly(P3, ATL, JFK)
9. Unload(C3, P3, JFK)
10. Load(C4, P4, ATL)
11. Fly(P4, ATL, SFO)
12. Unload(C4, P4, SFO)

Total steps: 12

Heuristics

Each heuristic has been tested with the first problem since it is the easiest. With this test as baseline it is possible to establish which heuristics will probably be too hard to used in a more complex problem. Those heuristics has been discarded for the Problem 2. The same process has been used for establishing which heuristics to test in Problem 3 using the results from problem 2.

In figures 1, 2 and 3 we can see the number of expansions, goal tests and new nodes for each problem.

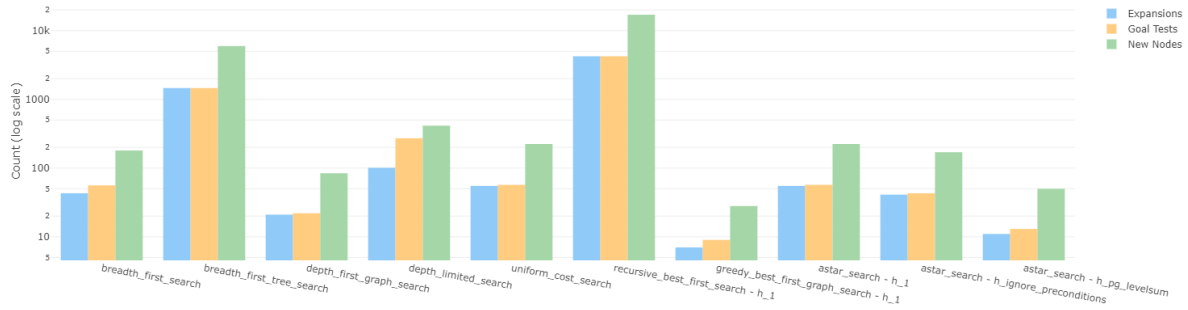

Figure 1: Number of expansions, goal tests and new nodes for Problem 1

Optimality

It is easy to see which heuristics are optimal and which not by comparing the number of steps needed by them to the steps calculated solving the problem by hand. In figure 4 we can see the number of steeps needed for each heuristic. If the number of steeps is greater than the minimum needed it is in red, if not, it is in green.

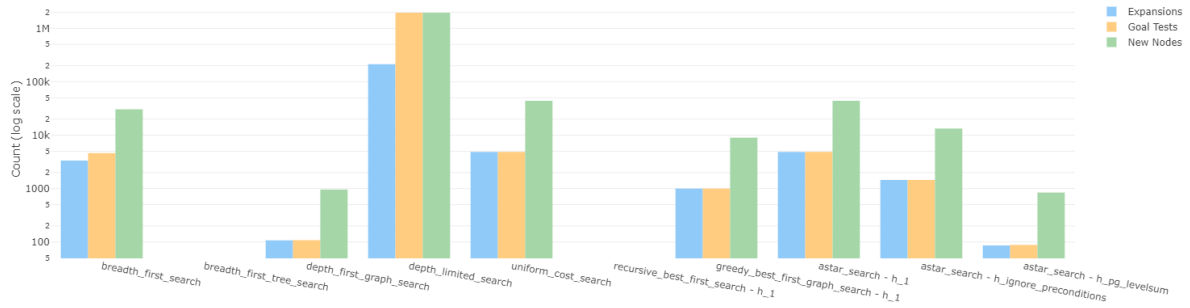

Figure 2: Number of expansions, goal tests and new nodes for Problem 2

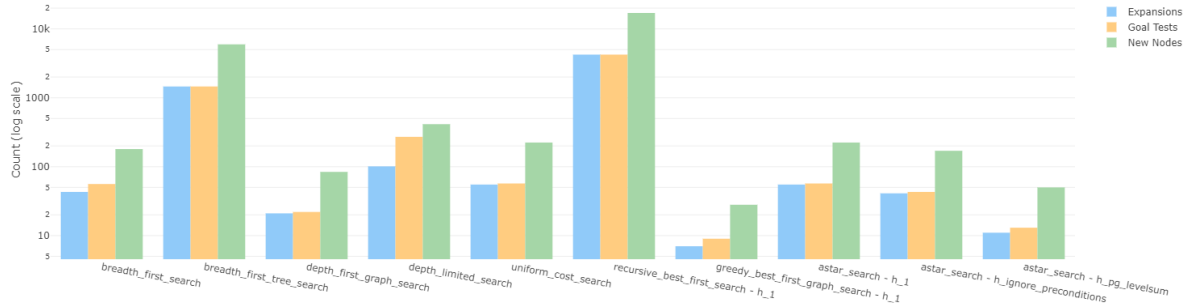

Figure 3: Number of expansions, goal tests and new nodes for Problem 3

It is import to highlight that greedy_best_first_graph_search - h_1 it is not optimal, it is only luck that in the first problem gets and optimal solution.

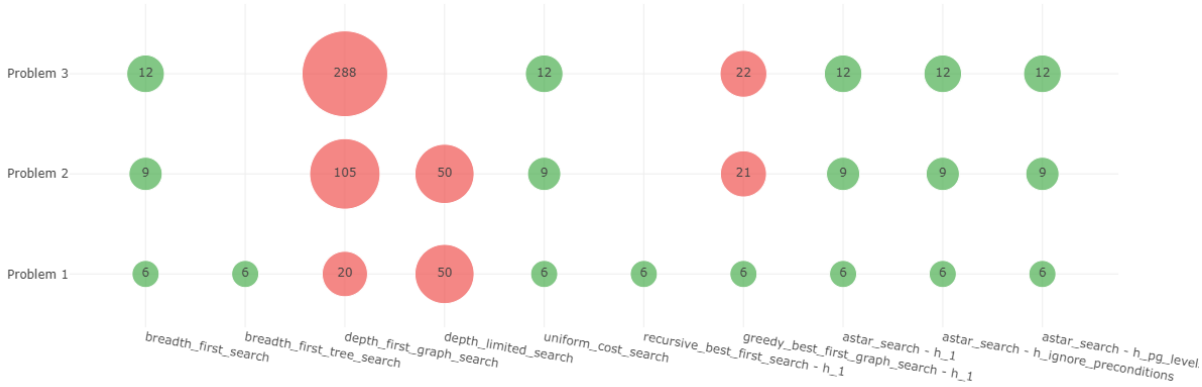

Figure 4: Plan length for each heuristic

Cost

For each heuristic it is also important to calculate the time needed to find the solution. Those can be seen in figure 5.

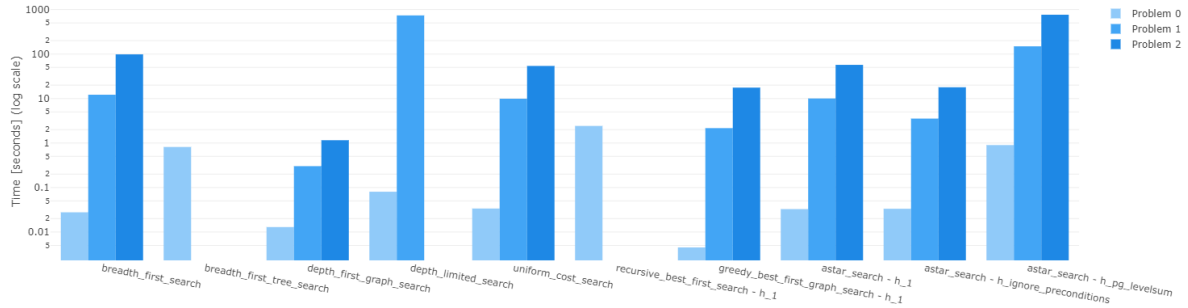

Figure 5: Time elapsed for each heuristic and problem

Best heuristic

The best heuristic is the one that gets and optimal solution in the minimum possible time. And this is astar_search - h_ignore_preconditions. By definition A* is guaranteed to find an optimal solution if the heuristics returns a value that is greater or equal to the real value. [1]

In the Search Lesson[1] they also say that a very good way to define heuristics is to take the problem and skip one restriction of the problem to obtain a relaxed problem. This is then used as to evaluate each node and it will guarantee that the solution found would be optimal. And this exactly what this heuristic is doing.

References

[1] Artificial Intelligence Nanodegree Lesson 11: Search. *Udacity.*