AIND Project 2
Forward-Planning Agent

Ruize Luo

November 11, 2018

1 Analysis of Results

1.1 Number of Nodes Expanded vs Number of Actions in the Domain

For the air cargo problem, there are three actions:

- $Load(Cargo, Plane, Airport)$
- $Unload(Cargo, Plane, Airport)$
- $Fly(Plane, Airport_1, Airport_2)$

Hence, given a problem with C cargoes, P planes and A airports, we can calculate the number of possible actions:

$$N = 2(C \cdot P \cdot A) + P \cdot (A \cdot (A - 1)) \tag{1}$$

And hence we can calculate the number of actions for each of the four problems:

Problem	# Cargoes	# Planes	# Airports	# Actions
1	2	2	2	20
2	3	3	3	72
3	4	2	4	88
4	5	2	4	104

Figure 1 is a plot of number of nodes expanded by each algorithm on each problem.

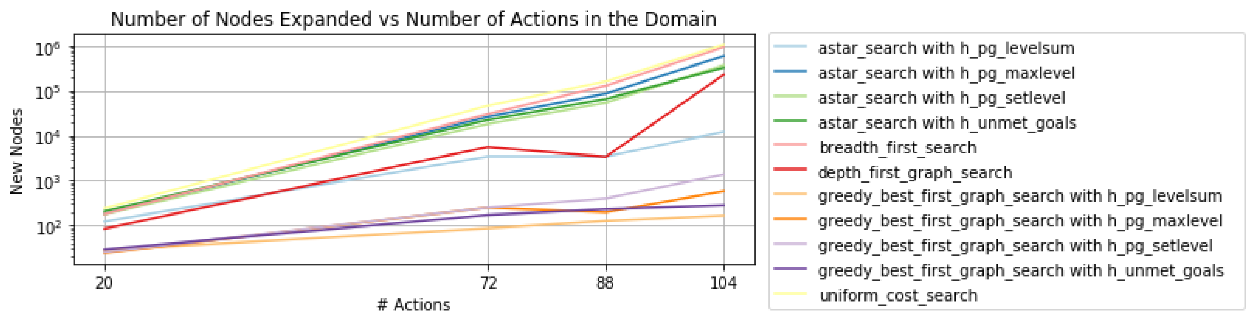

Figure 1: Number of nodes expanded vs number of actions in the domain for each search algorithm. Note that the y-axis is in log scale

From the graph we can see two things. Firstly, all algorithms show a log-linear relationship between number of nodes expanded and number of possible actions in the domain. The only differences among the algorithms are the slope of the line. Greedy algorithm seem to be looking into fewer new nodes than other algorithms.

Secondly, there is a dip in number of nodes expanded for depth first graph search (DFS) and greedy best graph search with maxlevel heuristics when going from 72 actions to 88 actions. This might be due to the fact that problem 3 only has two planes whereas problem 2 had three, since now at most two planes can have actions at each expansion. This would reduce the branching factor and thus making it easier for DFS to find the solution.

1.2 Search Time vs Number of Actions in the Domain

Below is a plot of search time for each algorithm on each problem.

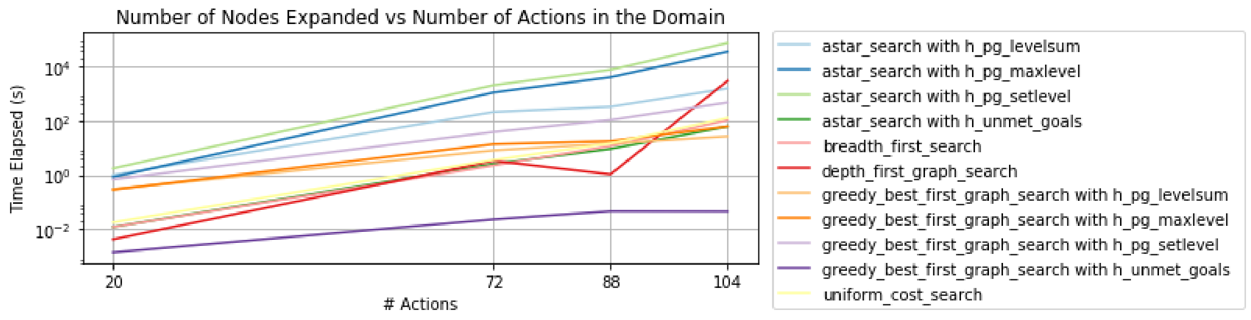

Figure 2: Search time (seconds) vs number of actions in the domain for each search algorithm. Note that the y-axis is in log scale

We can see that the relationship between search time is also approximately log-linear for all the algorithms. Greedy best first search with number of unmet goals heuristic outperforms other algorithms by being an order of magnitude faster. DFS has a steeper slope, which makes it suitable for very small problems (less than 20 actions).

1.3 Plan Length

Below is a plot of plan length for each algorithm on each problem.

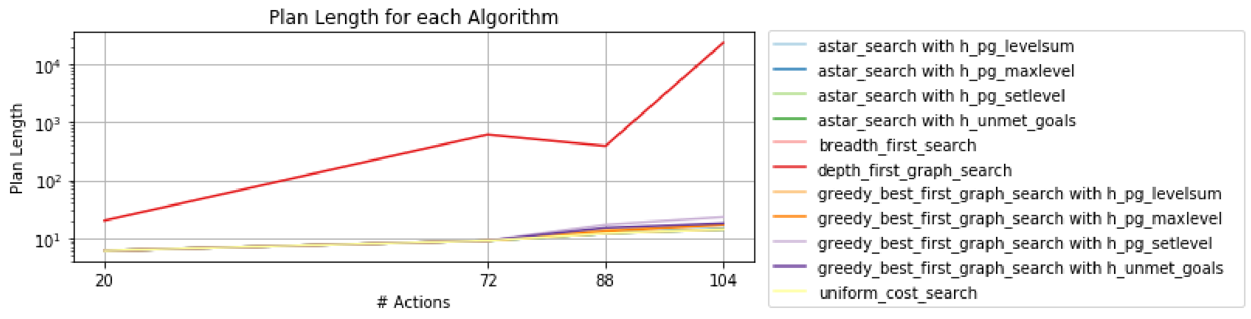

Figure 3: Plan length for each search algorithm on each problem. Note that the y-axis is in log scale

We can see that the path DFS plans is far from optimal and seems to be increasing faster than linear with number of actions even when the y-axis is already log-scale. In order to inspect the performance of the other algorithms, we should plot another one without DFS.

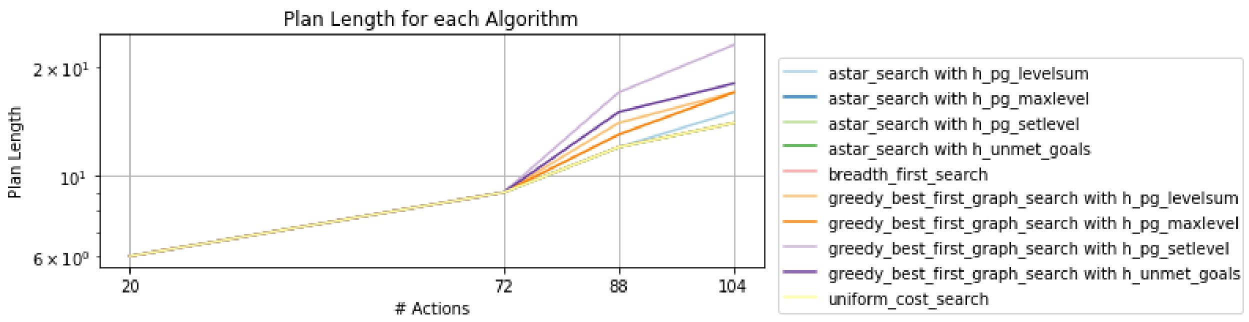

Figure 4: Plan length for each search algorithm on each problem without DFS. Note that the y-axis is in log scale

Note that there are 5 algorithms overlapping with the line that represents the shortest plan lengths for each problem:

- breadth first search (BFS)
- uniform cost search
- A* search with number of unmet goals
- A* search with maxlevel
- A* search with setlevel

We see that even though greedy best first search expands less nodes, A* searches with admissible heuristics find solutions that are optimal. Levelsum is not an admissible heuristic, and it did not find the optimal solution for problem 4.

2 Questions

1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

Answer: According to 1.2, DFS and greedy best first search with number of unmet goals heuristics operate the fastest under very restricted domains. However, DFS will become much worse if the domain ever expands.

2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

Answer: Under very large domains, the storage requirement of the algorithms start to play a factor. Hence, ideally we would like algorithms where number of nodes expanded grows slowly with the size of the domain. As can be seen from the above analysis, greedy best first search algorithms brings about the best storage efficiency, while still having reasonable run time and being able to find reasonably good solutions.

3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Answer: If optimality must be achieved, then A* with admissible heuristic, BFS and uniform cost search can be used (as listed in 1.3).