Udacity AI Nanodegree Project 2 Report

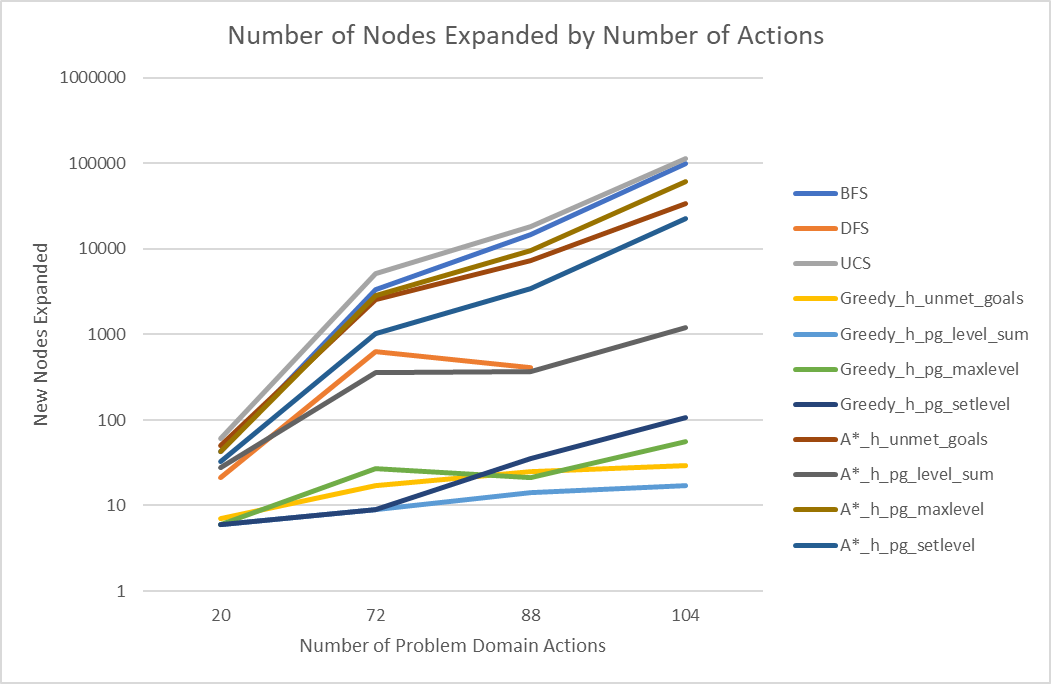
Kenny Carlsen

3/3/21

General Note: A plot of relevant data is provided in each section. Additional source data/plots are provided at the end of the report.

Note on algorithms executed: All algorithms were evaluated for all problems, however, due to an error in extracting the data for DFS on problem 4 the results for that particular algorithm/problem combination are not included here.

# Section 1: Search Complexity as a Function of Domain Size



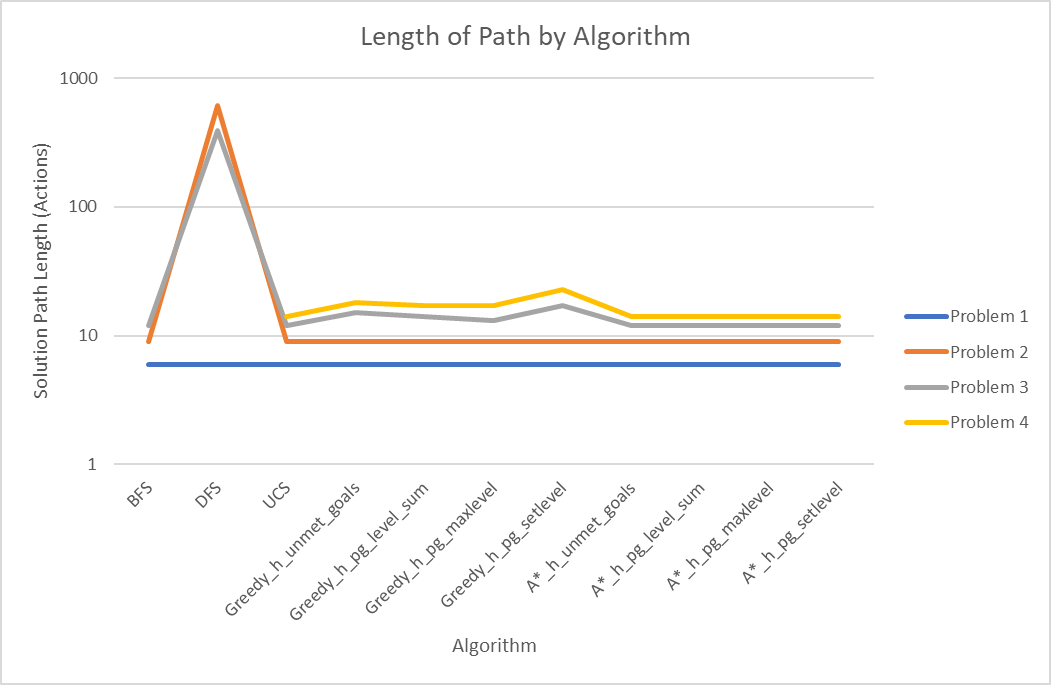
Note: Source data available in table form at the end of this report

In general across all greedy\_level\_sum performed the best in terms of number of new nodes expanded as a function of number of problem domain actions, with greedy\_setlevel performing comparably for “low” problem domain actions and greedy\_maxlevel and greedy\_unmet\_goals performing comparably for “high” problem domain actions. The results above appear linear on a log-lin plot indicating that these algorithms grow exponentially as a function of problem domain actions, but that the greedy algorithms grow at a slower exponential rate than A\* or uniformed counterparts. Interestingly, DFS performs better on the 88 action domain problem than the 72, likely due to the specific set of actions and goals in the domain which produce a favorable set of conditions for DFS performance.

# Section 2: Search Time as a Function of Domain Size

As in search complexity, greedy\_unmet\_goals again performs the best of the algorithms evaluated indicating that the problem space contains few or no ‘action traps’ which can cause greedy to sub-optimally meet goals to the point of a significant runtime penalty. It’s also interesting to note that the uniformed searches generally scaled better than the informed searches in terms of runtime (with the exception of greedy\_unmet\_goals).

# Section 3: Optimality of Solution by Algorithm



Most algorithms (excluding DFS due to its nature of running down tree branches searching for solutions before evaluating ‘simpler’, less deep solutions) performed comparably, reaching the same optimal answer in most cases. Notable exceptions to this behavior occur on the more complex problem spaces (problem 3 and problem 4) when using greedy algorithms. This is due to properties of the domain which, when decisions are made with greedy algorithms relative to the current position in a graph space, produce solutions which require more nodes than other solutions present elsewhere on the graph which stem from alternate nodes not currently under evaluation. Also note that the A\* algorithms consistently find the optimal solution across all domains at the cost of additional run-time and more node expansions (see questions 1 and 2).

# Section 4: Short Answer Questions

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?

Greedy algorithms (level\_sum in particular) would perform most optimally in a very restricted, real-time required domain due to their low node expansion, fast run-time, and their downside (potentially sub-optimal solutions) being mitigated by a restricted search space.

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)

Uninformed searches would perform well in large problem domains (specifically BFS and UCS) due to their ability to reach an optimal or nearly-optimal answer in a relatively short period of time.

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

A\* algorithms are the best choice when strict optimality is required as they perform optimally across all use cases evaluated at the expense of a longer run-time.

# Additional Source Data:

## Problem 1 Results



## Problem 2 Results



## Problem 3 Results



## Problem 4 Results

