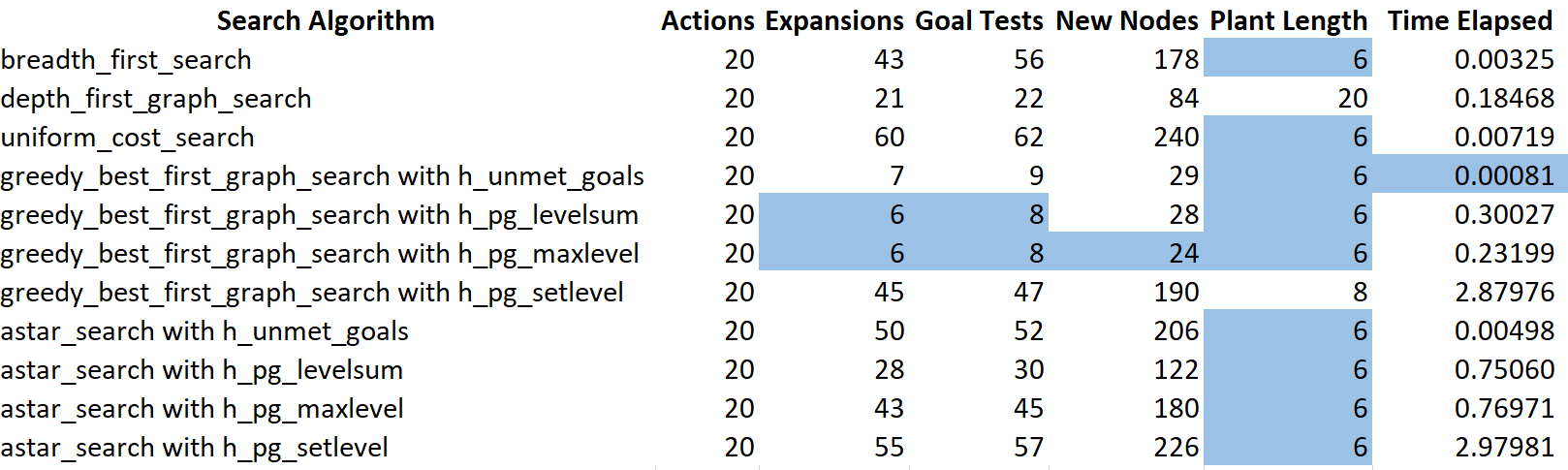
**Udacity Project 2 – Building a Forward Planning Agent Analysis**

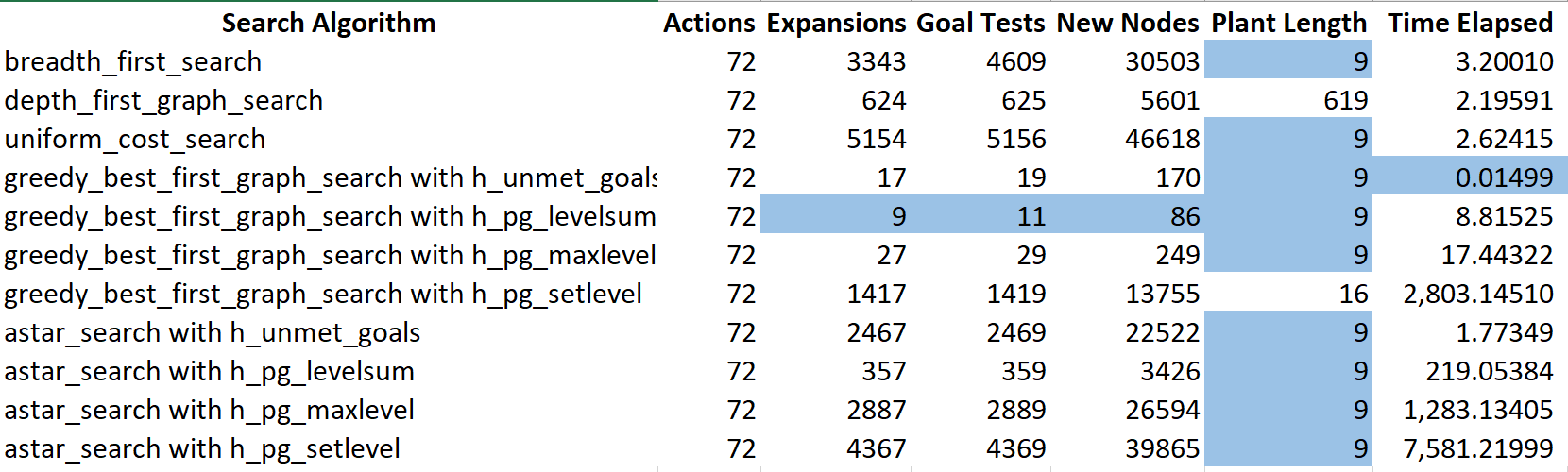
*By Sanuja Cooray*

I finished implementing the forward planning agent for the air cargo problems using uninformed search methods such as depth first, breadth first searches as well as informed heuristic-based methods such as A\* search.

**AIR CARGO PROBLEM 1**

****

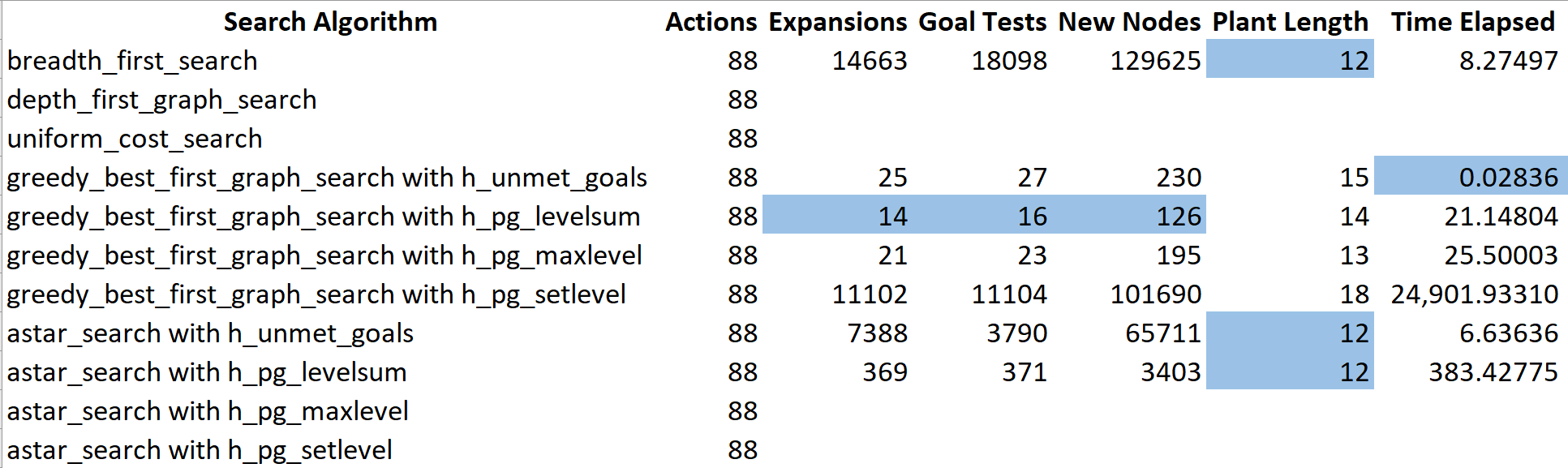
**AIR CARGO PROBLEM 2**

****

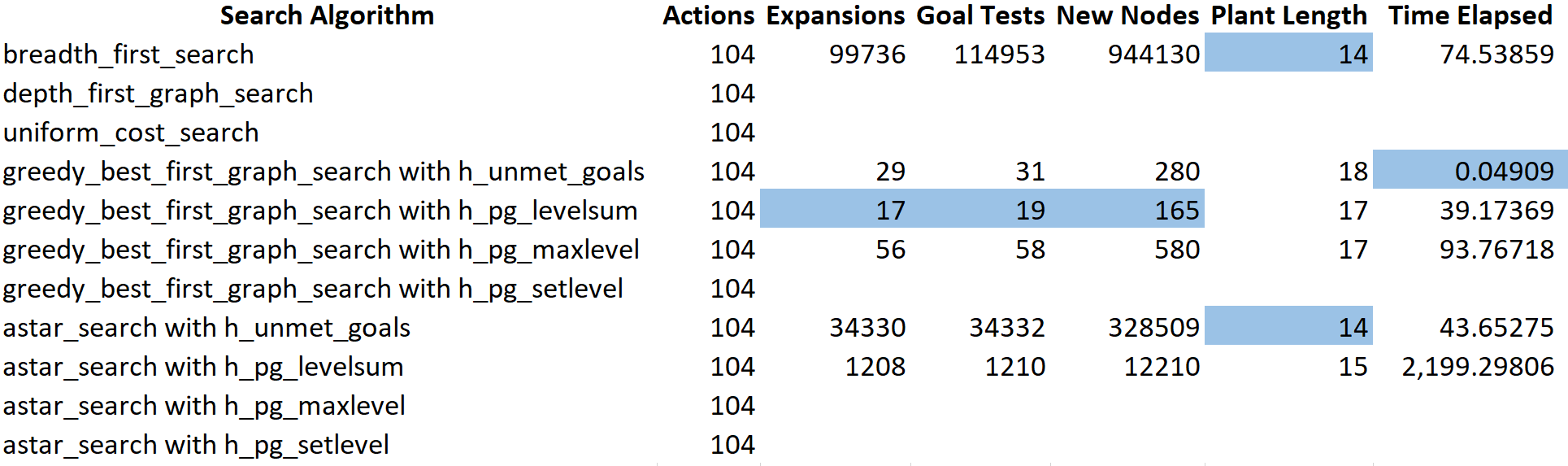
*Due to the higher plan length which requires substantially more memory and have very long execution times, I decided to exclude the following searches for the air cargo problems 3 and 4:*

* *uniform\_cost\_search*
* *depth\_first\_search*
* *astar\_search with h\_pg\_maxlevel*
* *astar\_search with h\_pg\_setlevel*

**AIR CARGO PROBLEM 3**

****

**AIR CARGO PROBLEM 4**

****

**ANALYZING SEARCH COMPLEXITY**

As the complexity of the problem increases, the search complexity rises across all algorithms however it does not rise equally across all algorithms. Algorithms such as depth\_first\_search see an exponential increase in search complexity as the plan length increases dramatically in comparison to other uninformed searches.

However, in other uninformed searches, the number of nodes explored, and the expansions made rises dramatically as the problem complexity increases. The same phenomena is observed in the A\* search algorithms as well where the search complexity increases dramatically, this similar relationship could be due to the shared attributes of breadth\_first\_search and A\* searches even though the A\* search has the additional layer of using a heuristic as well.

Fascinatingly, the algorithm that is purely heuristic based, namely the greedy\_best\_first\_search performs the best with the least increase in search complexity as the problem complexity increases.

Considering the heuristics used across both A\* and greedy\_best\_first\_search, it is clear that h\_pg\_levelsum and h\_pg\_setlevel showcase the least increase search complexity. I believe this is due to the increased accuracy of the heuristics in related to the problem.

**ANALYZING TIME**

As the complexity of the problem increases, the time taken to solve the problem increases across all algorithms however it does not increase equally across all the algorithms and all heuristics.

When considering the algorithms, greedy\_best\_first\_search once again performed the best with the least increase in time and least time taken to complete the solution.

Algorithms that used heuristics such as h\_pg\_setlevel and h\_pg\_maxlevel observed an exponential increase in time taken whereas in relation, heuristics such h\_unmet\_goals saw the least increase. I believe, h\_unmet\_goals performed the best because other heuristics simply take too long to calculate, although the accuracy of those heuristics maybe better, the time take to calculate the heuristics simply makes them infeasible as the complexity of the problem rises.

**ANALYZING OPTIMALITY OF THE SOLUTION**

I would personally base the optimality of the solution simply based on resource utilization specifically memory consumed which is linked to increased search complexity and most importantly time taken.

In both cases the pure heuristic based method of greedy\_best\_first\_search comes out on top with the least increase in search complexity hence least increase in memory utilization while also maintaining the least increase in time.

When evaluating the heuristics, h\_pg\_levelsum and h\_pg\_setlevel have the least increase in search complexity with increasing problem complexity however I do not see them as the ideal solution simply due to the exponential increase in time consumed due to the complexity of calculating the heuristic. Since I prioritize time consumed over memory consumed, I would say the heuristic of h\_umet\_goals is fair better due to the substantially less increase in time consumed.

**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?

Considering that real time applications require real time feedback, I would prioritize execution time to be the most important hence for such a situation I would use the greedy\_best\_first\_search algorithm with h\_unmet\_goals heuristic.

1. 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)?

Since the time restriction is a lower priority here, I would use A\* search combined with a good heuristic function. I would still pick the h\_unmet\_goals function here simply due to the complexity of the problem, the other heuristics will simply take far too long to calculate thereby making infeasible.

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

Since the most optimal plans are the final goal and if time consumed and memory utilized are not important. I would simply use and uninformed search method specifically breath\_first\_search or uniform\_cost\_search to find the most optimal plan as these algorithms are guaranteed to converge on the most optimal plans due to thorough search of the problem space in comparison to other algorithms which maybe be biased with the use of heuristics and end up with suboptimal plans. However, as the problem complexity rises the requirement of resources to solve this problem also exponentially rises.