**Report for Project 2: Classical Planning**

**Yudai Furukawa**

**Analysis**

1. The number of nodes expanded against number of actions in the domain
   1. According to table 1 and chart 1, as the number of actions increases, the number of nodes expanded increases in general. For example, for breadth first search, the number of nodes expanded exponentially increased from 43 to 99736 as the number of actions increased from 20 to 104.
2. The search time against the number of actions in the domain
   1. According to table 1 and chart 2, as the number of actions increases, the search time increases in general although the rate of increase depends on algorithms. According to chart 2, the search time of astar\_search with h\_pg\_maxlevel increased the fastest among the algorithms as the number of actions increased.
3. The length of the plans returned by each algorithm on all search problems
   1. According to table 1 and chart 3, as the number of actions increases, the length of the plans increases although the rate of increase depends on algorithms. The rate of increase is similar among algorithms besides that depth\_first\_graph\_search has higher rate of increase compared to other algorithms.

**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?
   1. The most appropriate algorithms for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time would be blind search algorithms (depth\_first\_graph\_search, depth\_first\_graph\_search, and breadth\_first\_search) as there is no need for a heuristic given the domain is very restricted.
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)
   1. When domain is big and there is modest time constraint, greedy\_best\_first\_graph\_search with h\_unmet\_goals seems to be the most appropriate among the algorithms. The algorithm is not guaranteed to find an optimal solution. However, the algorithm with a good heuristic can find a solution to a given problem without exponentially increase search time in very large domains.
3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?
   1. When it is important to find only optimal plans, breadth\_first\_search and uniform\_cost\_search would be the most appropriate algorithms as the algorithms are guaranteed to find optimal plans. A\* search would be a good candidate as well since it finds optimal solution to problems as long as the heuristic is admissible which means it never overestimates the cost of the path to the from any given node.

**Algorithms excluded from the test on Problem 3 and Problem 4**

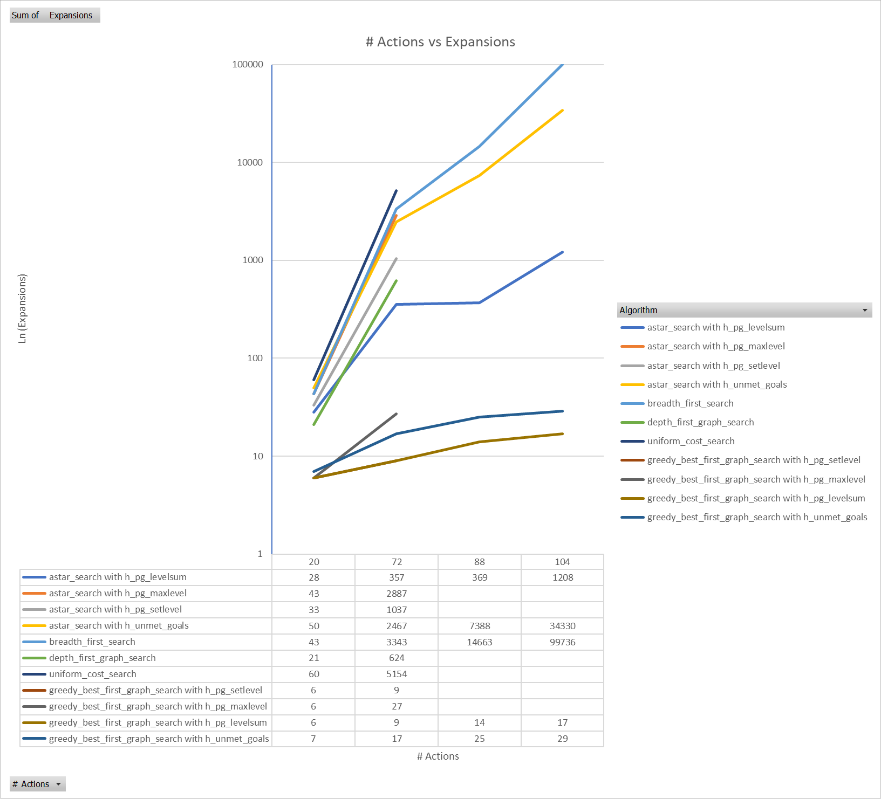
For Problem 3 and Problem 4, I excluded the following as Time elapsed in seconds exponentially increases as the number of actions increases.

|  |
| --- |
| * uniform\_cost\_search |
| * depth\_first\_graph\_search * greedy\_best\_first\_graph\_search with h\_pg\_setlevel * greedy\_best\_first\_graph\_search with h\_pg\_maxlevel * astar\_search with h\_pg\_maxlevel * astar\_search with h\_pg\_setlevel |
|  |

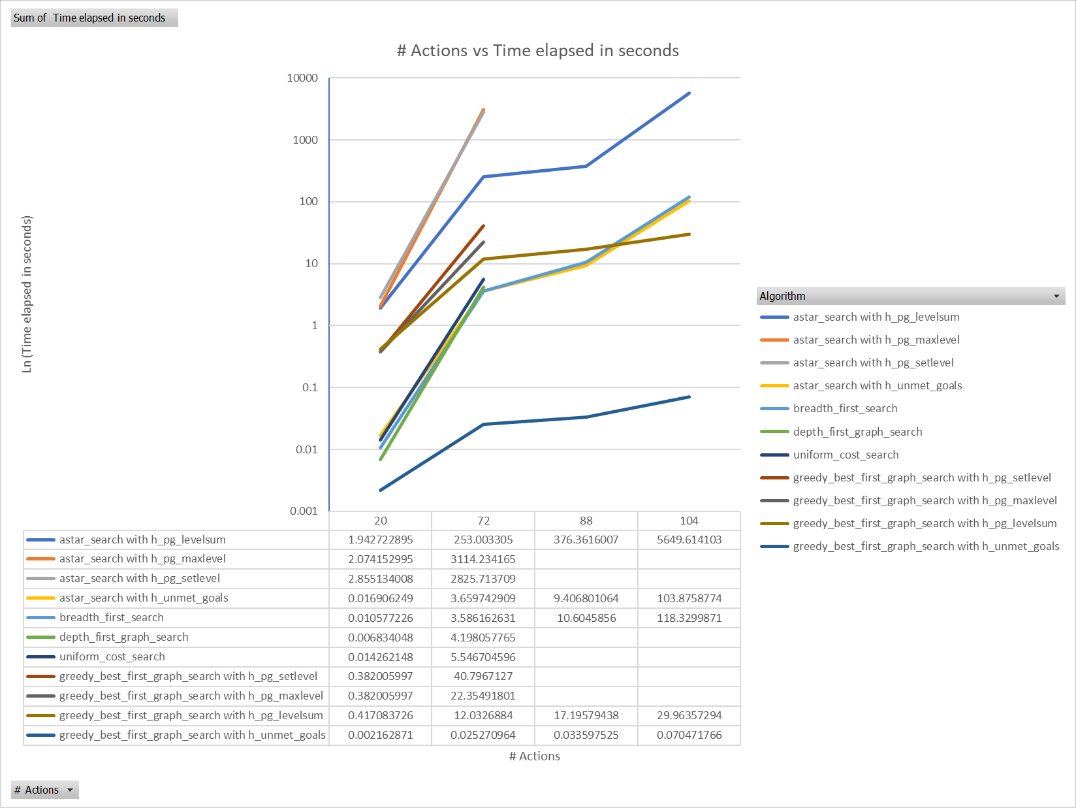
[Table1: Results]



[Chart 1: # Actions vs Expansions]



[Chart 2: # Actions vsTime elapsed in seconds]



[Chart 3:Plan Length]