CS 473 Final Project Algorithm Analysis

Group Members: Andrew Hutson, McKenna Galle, Vina Le

# Empirical Analysis

Our original number of paths was 14 factorial (87,178,291,200). We increased the number of paths to 28 factorial (~3.05\*1029). The runtime for 14 factorial was 30 milliseconds (as measured by the Visual Studio 2017 debugger), and 38 milliseconds for 28 factorial (again, measured by the debugger). From this data, we calculated a ratio of ~1.267, which is an increase of ~26.7% of run time.

# Theoretical Analysis (Worst Case)

By using Branch and Bound, the weights of all nodes were inserted into the program by hand with no deviations. With that, there exist a convex optimization that assigns the first node (facilities on Whitworth’s campus) as the starting point *z* where *z1* to *zn* is carried out in exhaustive search. By having the program being non-heuristic, the relations of z was put in terms of a Boolean vector – eliminating the largest upper bound and set the lower bound – then checks for all previous nodes in the paths and eliminating that node. The worst case theoretical number of paths of our branch-and-bound algorithm is n\*2n. For 14 nodes, the worst-case number of paths is 229,376 recursions - assuming the correct path is found last every time. For 28 nodes, the worst-case number of paths is 7,516,192,768 recursions. This yields a ratio of 32,768, which is an increase of 3,276,700%.

# Theoretical vs. Empirical Comparison

As you can see, there is a considerable difference between the empirical data and the theoretical data. We believe this difference is due to our algorithm not running the worst case. We’re pretty sure that the program is running somewhere in-between the best-case and the worst-case efficiency.

By using theoretical, the program needs to execute every possible path per building although empirical takes the given paths and eliminates all paths that included nodes (or buildings) already taken. While the empirical doesn’t need to go to every path because it only calculates the shorter paths.