**Dataset Sizes**

Forest\_D\_ALL - 18 species X 29 resources - 522 2n = 262144  
Forest2 - 46 species X 39 resources - 1794 2n = 7.03x10^13  
Forest1 - 89 species X 37 resources - 3293 2n = 6.19x10^26

**Benchmarking Tests**

Original Software - Full output  
Forest\_D\_ALL - ~300ms  
Forest2 - ~10400ms  
Forest1 (50m iterations) - ~120400ms

Improved software - Full output Truncated output  
Forest\_D\_ALL - ~300ms ~260ms  
Forest2 - ~10600ms ~2200ms  
Forest1 (50m iterations) - ~62000ms -

**Translation to C++**

The Java code was initially translated to C++. This version of the software is fully functional but runs approximately 100 times slower than the Java code. This is likely due to lack of housekeeping/garbage disposal and use of vectors that are not preallocated with a size. C++ abandoned in favour of further investigation in Java.

**Algorithm improvements**

Avoiding multiple unnecessary calls to getFirstNonCompleteLeaf in while loop of getMinimalConstSpecTreeRootedAt() method

parallelStream replaced nested for loop in getMinDomSpecSets() (Can’t be used with multi-threaded GUI)

Sorted output species in number order (has a time penalty but important for consistent comparison of output between software versions)

**Discovery**

The problem with solving for Forest1 is not essentially the dataset size but due to problematic species within the dataset that dramatically increase the MinimumSpecSetFamily size. The problem species tend to be ones that use many resources but this is not an exact rule. Removal of 10 of the 89 species in Forest1.txt allowed solving in around 4 hours.

**GUI**

A GUI was developed to allow data file selection and process with or without truncated results. Truncation allows the software to stop after 3 consecutive increases in Minimum Mean Sensitivity and saves much processing time. A plot of minimum mean sensitivity is shown on the GUI.

A second part of the GUI is in development to automate the process of identifying problematic species. This is progressing well and is now in the testing and refinement phase. Currently, this processes subsets of a dataset in series but could be extended to make use of the cluster. This will be fairly tricky to implement.