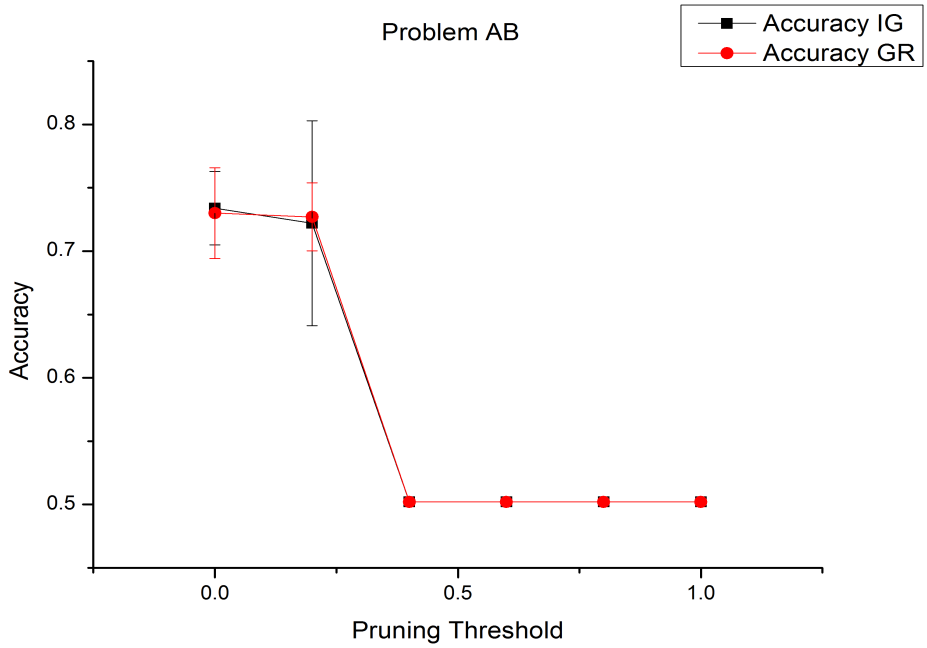
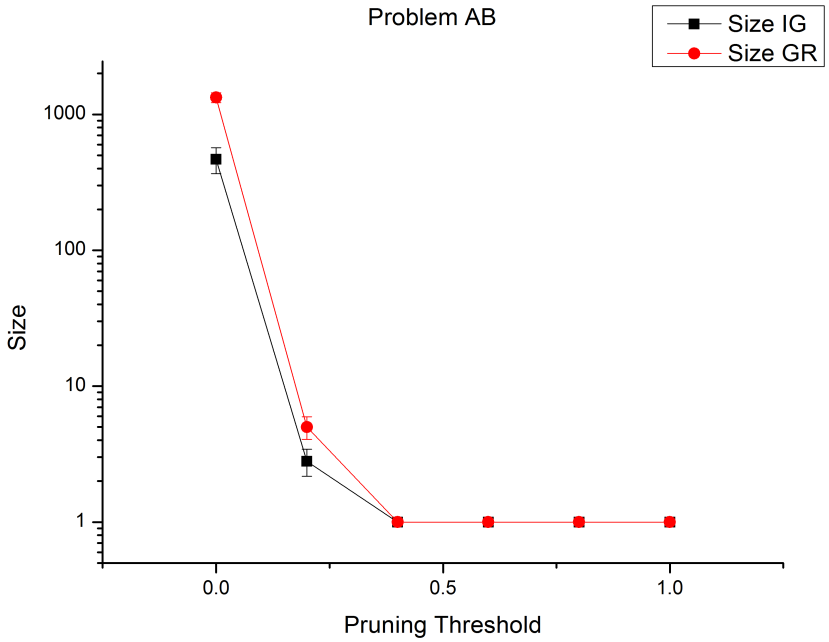
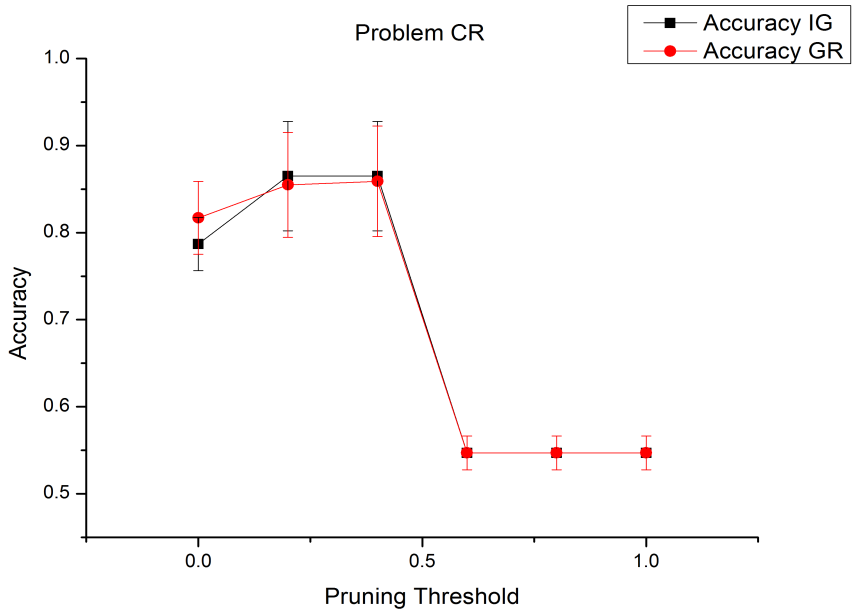
Eric Perko
exp63
EECS 440 PA 1 Writeup

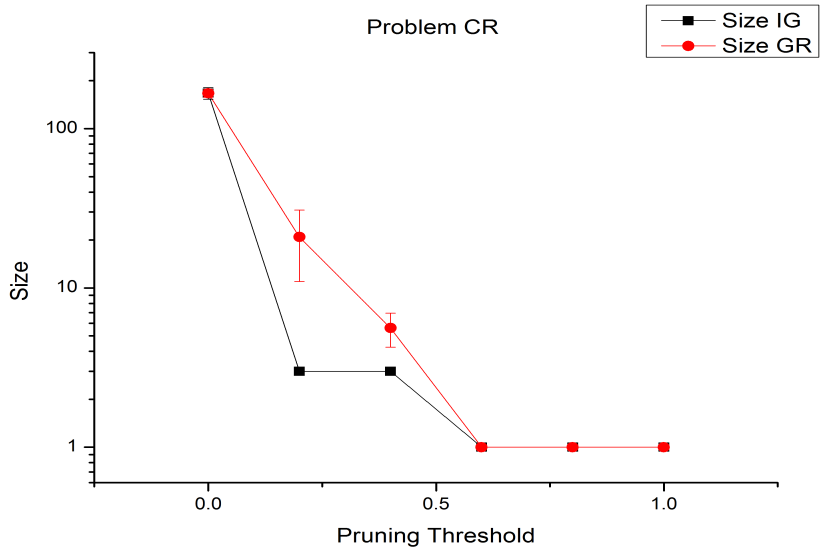
A)



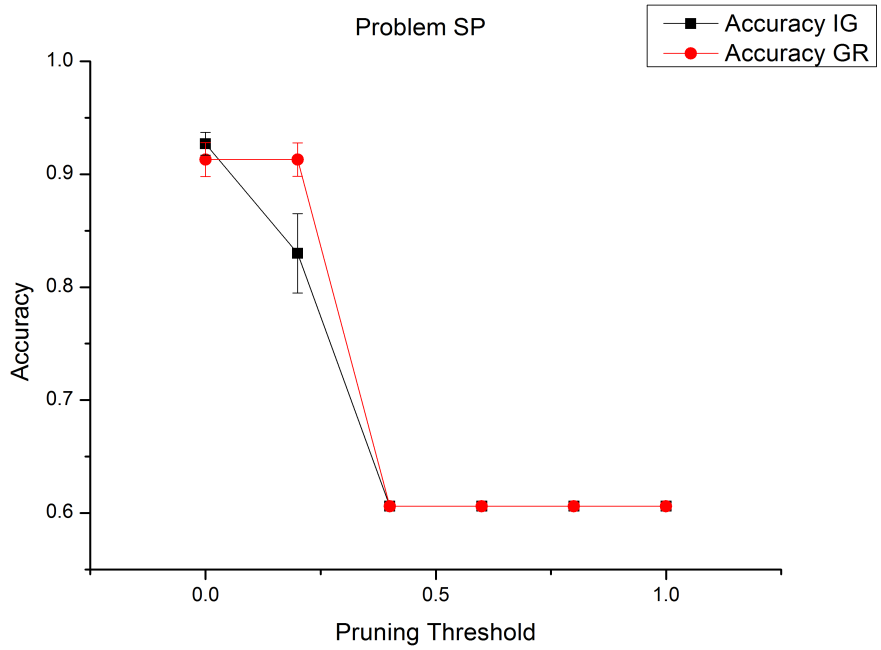


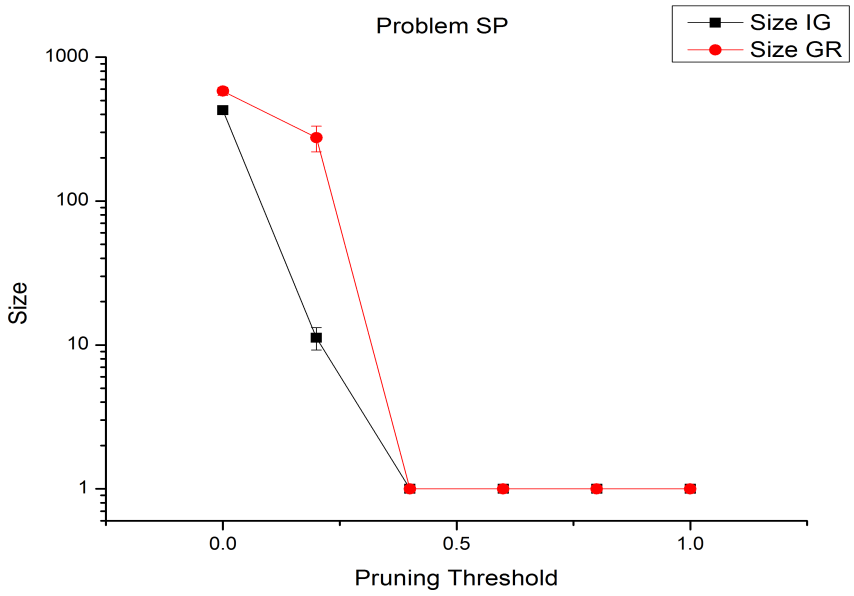
Note the logarithmic scale on all size graphs. No pruning yielded the most accurate graphs for problem AB, but generates a very large tree. A pruning threshold of 0.2 yielded a much smaller tree and only slightly worse accuracy. Further testing showed that the sudden drop in accuracy happens just after 0.2, most likely due to the fact that after 0.2, we stop returning test nodes and the tree starts guessing at the root.



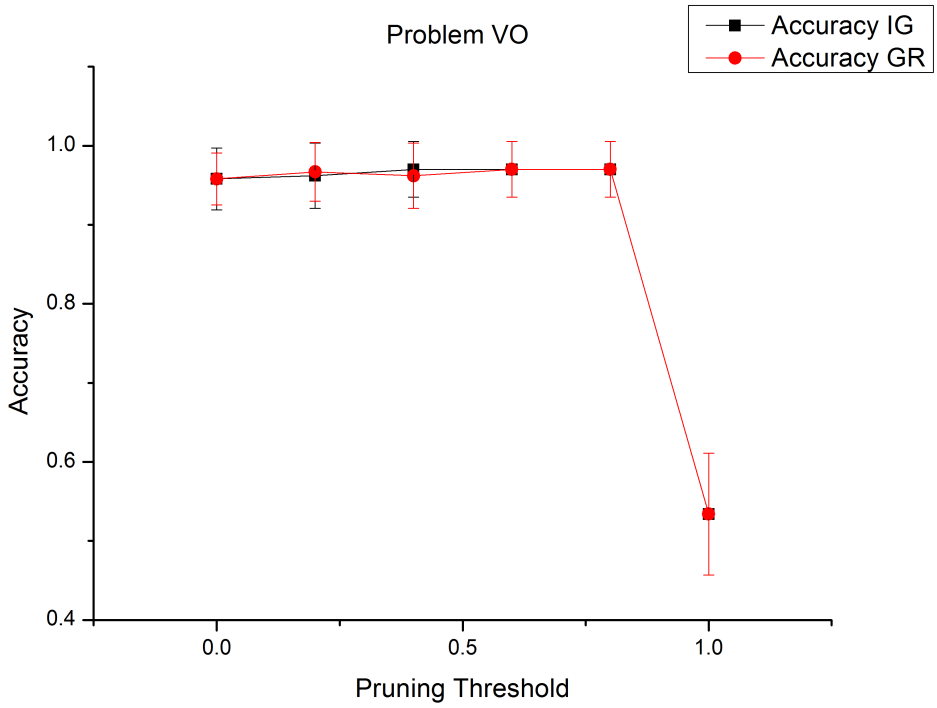


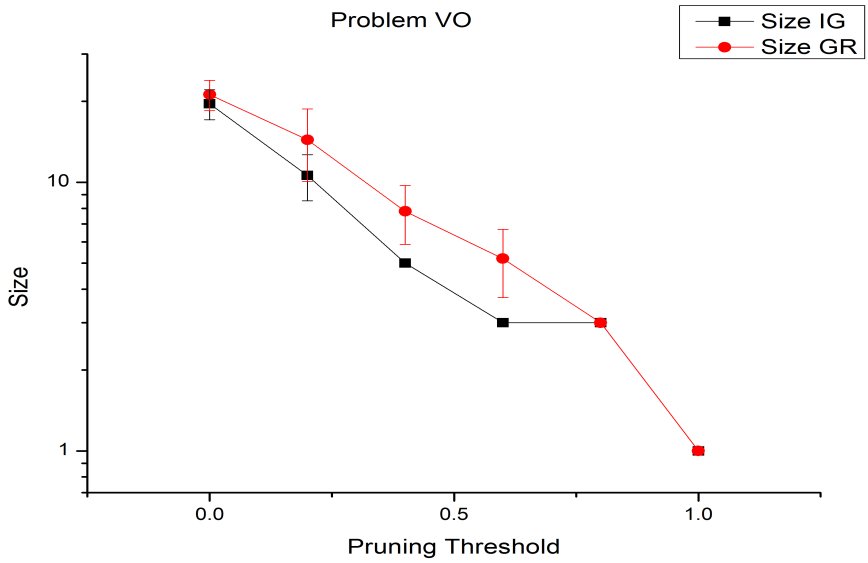
A pruning threshold of 0.4 gave the best accuracy and size for CR. Unlike most of the other problems, increasing the pruning threshold increased accuracy for both split criterion.



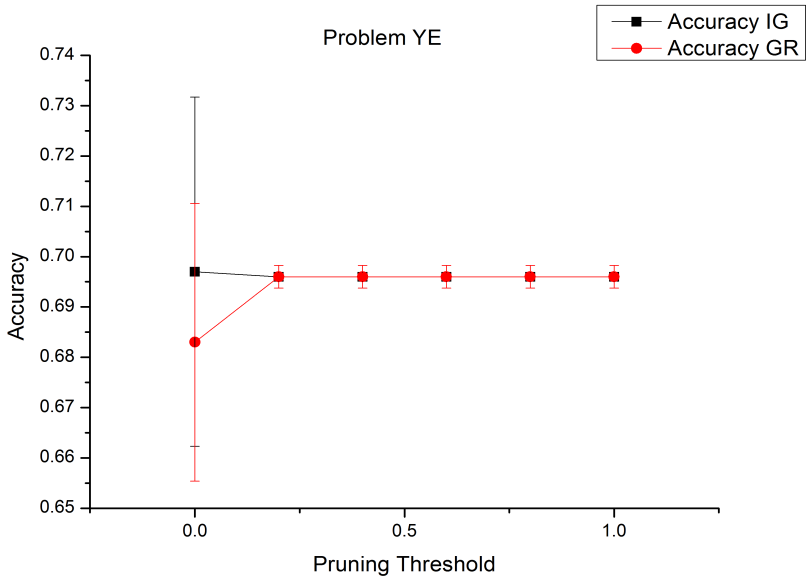


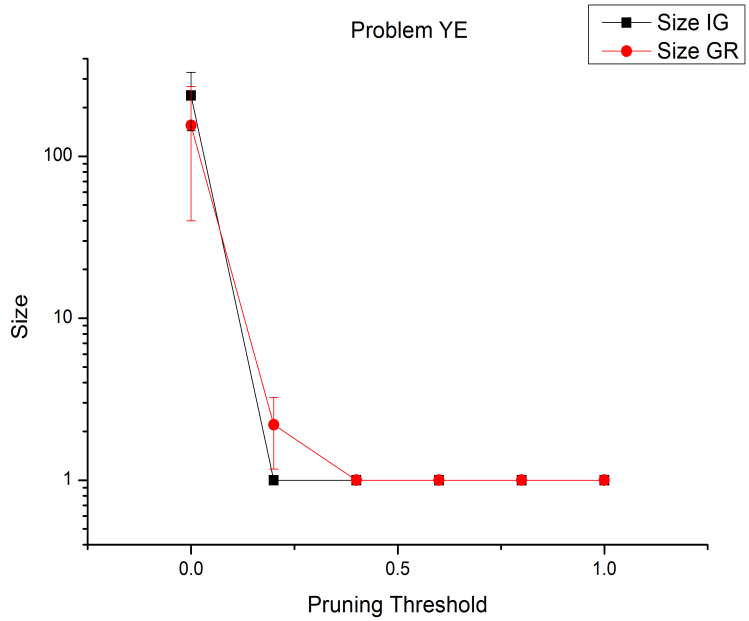
No pruning threshold yielded the best accuracy for SP. It also yielded a very large tree. However, for SP, a decrease in tree size also caused a drop in accuracy, especially for regular information gain.





For VO, a pruning threshold of 0.6 was most effective. The accuracy of the tree is the same (within error bars) as with no pruning, but the size of the tree was smaller than any lower thresholds.

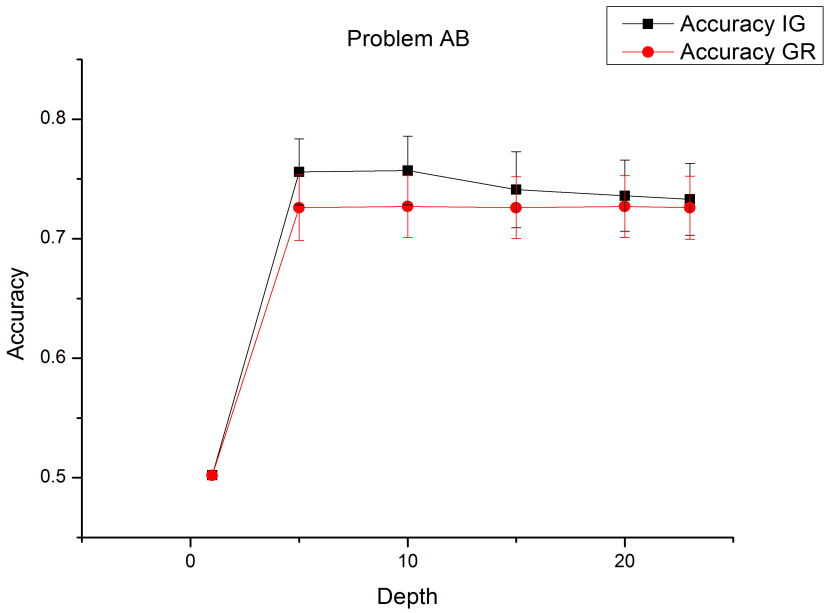


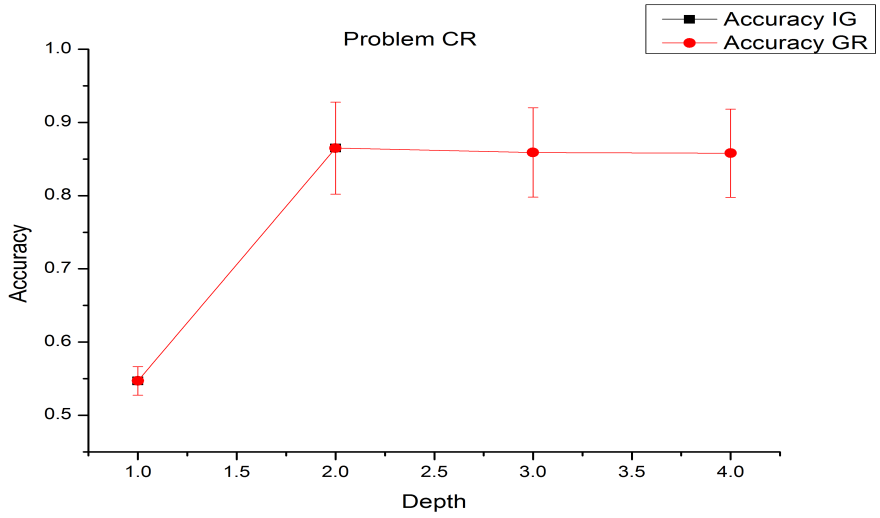


Any pruning thresholds greater than or equal to 0.2 is best for this problem. However, no threshold actually generated a good decision tree. The accuracy the tree achieved corresponds to a weighted accuracy of about 50%, and the tree performed as well with nodes as at a thresholds of 1 which returns an empty tree. This implies that the target concept in YE is not learnable by a decision tree and cannot be learned by a linear classifier such as a decision tree.

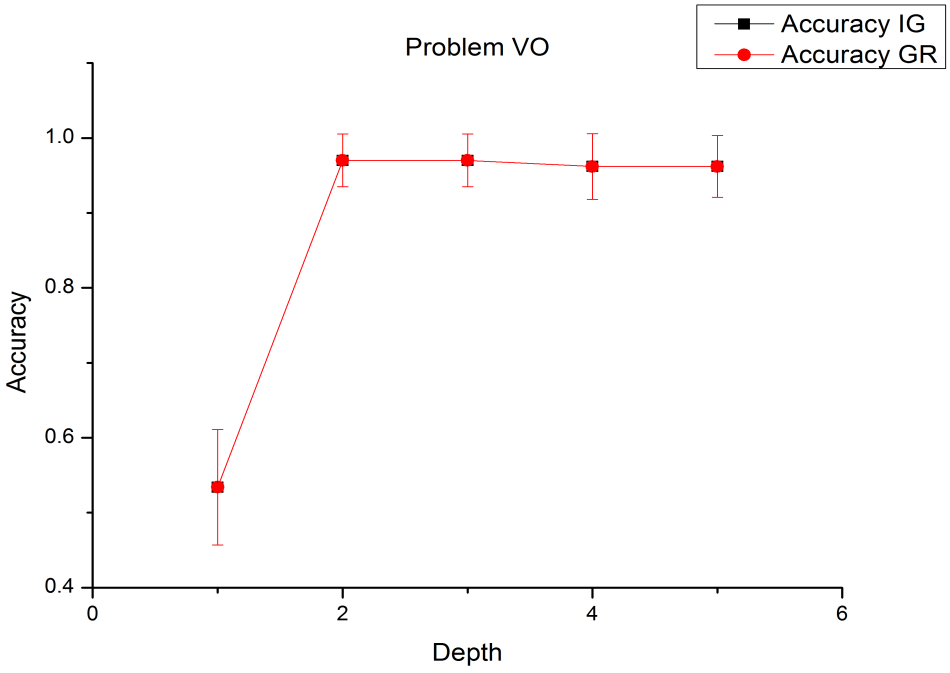
On average, gain ratio was a bit better than information gain as a split criterion. The only problem where it did not match or exceed information gain was YE, which I do not believe is a linearly separable problem.

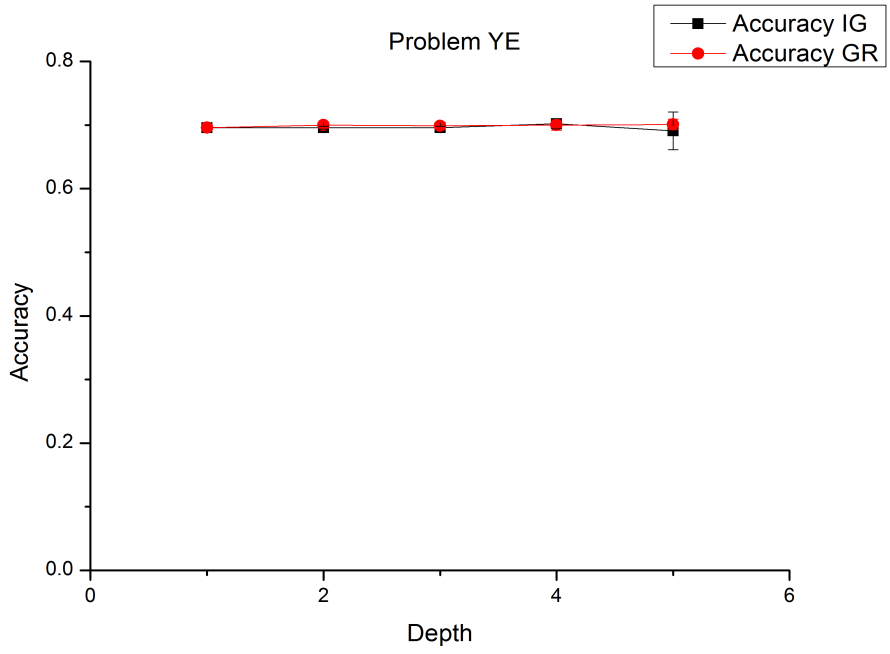
b)

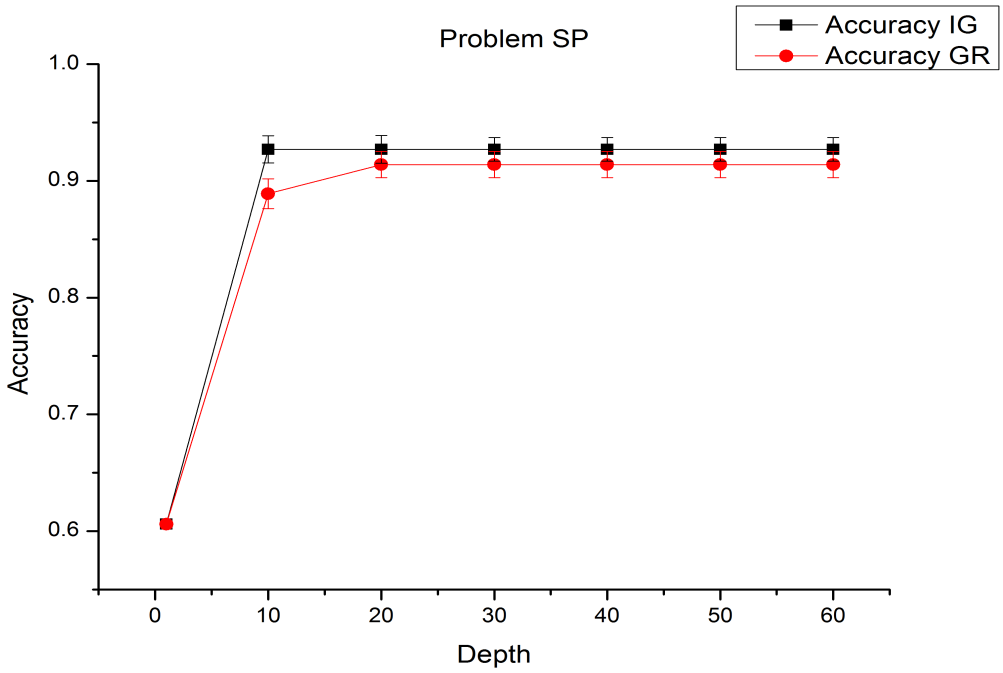




Note: For CR, using IG as split criterion caused the program to crash and I was unable to find the cause. This occurred at a depth above 2, but the optimal depth from part A was actually 1 for this problem, so I would not have expected the accuracy to increase anyways.


Problem VO
Accuracy IG
Accuracy GR
Accuracy
1.0
0.8
0.6
0.4
0
2
4
6
Depth


Problem YE
Accuracy IG
Accuracy GR
Accuracy
0.8
0.6
0.4
0.2
0.0
0
2
4
6
Depth



None of the classifiers seemed to linearly climb to their optimum depth. The climb looks much more like a logarithmic function, which makes some intuitive sense – adding a node early in the tree is more likely to improve accuracy than adding a node towards the end of a long branch.

c) For problem AB, the tree is composed of only two tests. The root performs a test on the height of the example (I couldn't determine exactly what the separating value it chose was). The tree then tests on for the sex. These are the only two tests. These do seem like reasonable tests, as the height generally increases with age, and sex will tell narrow it down, as each sex grows at different rates and has different average heights for a given age.

For problem CR, only 1 test is performed – a test on attribute A7. I cannot determine if this is reasonable, because the info file doesn't actually state what A7 represents in terms of the credit approval process.

For problem SP, I was unable to determine the tests it performed due to the long running time of the problem.

For problem VO, the tree only tested the water project cost sharing attribute. This does seem like a good predictor of political party, as the two parties generally have differing (and not overlapping) ideas when it comes to how to pay for government projects.

For problem YE, the tree does not perform any tests to achieve maximum accuracy. It is blindly guessing the majority class label of the training set.

General thoughts:

Mostly, I noticed that the running time of these decision trees is much longer than the running time of the decision tree I implemented last year for EECS 491 (the SP problem can take a few hours without any pruning). While some of this may be due to implementation differences or Python vs. Java differences, it seemed to me that introducing continuous attributes greatly increased the amount of time spent calculating max IG/GR. Some of the problems had a very large number of possible separating values, all of which needed checked. This increased the running time a lot vs. data sets such as VO that were completely binary.