Project 2A

Jordan Lynn

09/28/2015

My project was influenced by Dr. Terry Soule's code found here `http://www2.cs.uidaho.edu/~cs472_572/f15/GPProjectA.html` while not an exact copy, some of the program flow was referred to. This project is also in C++ so this made things go faster and more smoothly.

1 Individuals

1.1 Generating an Individual

The "Individual" class is capable of generating trees for with operators $+, -, *, \div$ these operators are referred to as integer constants within the code. The numbers are defined in the code as follows.

```
#define add 0
#define sub 1
#define mult 2
#define div 3
#define terminal 4
#define constant 5
```

When a individual is generated it constructs a Binary Tree (BT) to a given depth (for this project is was a random number between 1 and 3 although I believe this will need more fine tuning later.) that is defined when the function is called, building a child node branch until it arrives at the desired depth then working it's way back up building the children to the right, a rand()%5 allows for a random assignment of operators/terminals unless the node being assigned is a leaf, then simply a terminal or a constant is inserted. A switch statement decides if the node needs a constant value, a terminal, or a operator and will take the appropriate steps to fill that specific node. So the individual is delivered with a full Binary Tree (BT) that has been given a random assortment of either terminals or non-terminals based off of the definitions from above. Once an Eval() statement makes it's way to a node a switch() statement handles arithmetic operations based on the #defines from above it simply follows down the left then right children of the tree.

The individual class also has and Eval() function and a print() function. The Eval() function will apply the numeric possibilities to the function and will also filter the solution through a root, mean, square error analysis to help the individual's tree stay on track as discussed in class.

2 Results/Conclusion

Initial runs show some progress in the beginning but quickly plateau, as in coming nowhere near the correct solution. This may be fixed by creating some fitness function that favors smaller trees, but more testing is needed. The program would quickly cause the trees to grow to huge sizes which slowed down the computer working on the solution. The following graph shows how the algorithm struggles with an initial x^2 plot but then is more able to find points for a random assortment of numbers.