CS572 Project 2a Report

Heyan Huang

March 7, 2014

Contents

1 Algorithm Descriptions

I built my algorithm from a bottom up approach, starting from nodes, and then individuals, and then a population with size of individuals of 100.

2 Individual Descriptions

2.1 Data Structure

I have use the tree structure to build my individual. To build a valid tree individual I will pass in the max depth of the tree, and then create the tree. When creating the tree, I started from build a NULL root node, according to node type, terminal, or non-terminal, building the subtree nodes with the 3rd and 4th branches of the parent node set to have null value.

For each node, I have a node pointer points towards the parent, and four node pointers point towards all the four chicdren.

2.2 Individual Terminal Nodes

I have two type of individual terminal nodes, the inputX and constant values. InputX reads the value from corresponding points, and constants values are the randomly generated floating point values.

2.3 Individual Non-terminal Nodes

For this project, since I have started late, changed from trying to use Java, I just did the most basic four non-terminals: +, -, * and /.

2.4 Copy Individual

To copy individual, I did create a individual object first, and then created an node object as the root of new individual tree. I copied all the information of the root node from the source, and then I set the the$_{indiv}$ pointer points to this root node. Then all I need to do is just call the node copy function by passing in the the$_{indiv}$ pointer as the argument to finish all the rest of copying.

```
Individual Individual::copy()
{
    Individual newtree;
    newtree.fitness = fitness;
    newtree.size = size;
    newtree.terms = terms;
    newtree.non_terms = non_terms;
    node root;
    root.type = (*the_indiv).type;
    root.const_value = (*the_indiv).getVal();
    newtree.the_indiv =  &root;
    root.copy(the_indiv);
    return newtree;
}
```

In order to be able to visualize that I have conducted the copy correctly, I write a print tree function to print my individual tree and also the type associated with each node so that I can compare.

2.5 Evaluate, Erase individual

the evaluate individual function and the erase individual function are pretty much the original codes that Dr. Soule has given to us, don't need much change yet. They work just fine.

2.6 Calcuate size

Since all the node, individual classes have the size function defined already to get the size of each tree, and we could also update the size of the tree by applying the calc_{size} function, and for me for this subproject, I applied fixed max depth of tree of depth 5, so the sizes of all the individual are fixed to be 63. Not much work needs to done here as well.

2.7 Fitness function

with the limited time i have right now, I didn't care too much about the fitness so far.

2.8 Population of a fixed size

Just like project 1, the population class was there already, all I need to do is just copy the necessary functions to make this minimumly required population be able to do the things that needed.

The associated functions that have applied on this class including the following:

2.8.1 minFitness

Loop down all the individuals in the population and calculate the fitness for each individual, records the mimimum fitness value within the population.

2.8.2 avgFitness

Loop down all the individuals in the population and calculate the fitness for each individual, records the sum of all fitnesses, and finally return the floating value of sum divided by the size of the population.

To show an example of these three functions, I have pasted the code I used for this function:

```
float Population::avgFitness() {
    float avgFitness = 0.0;
    for (int i = 0; i < size; ++i) {
        avgFitness += popu[i].fitness;
    }
    avgFitness = (float) (avgFitness / size);
    return avgFitness;
}
```

2.8.3 avgIndiSize

The idea is that when I generate random max depth individuals, the individuals have different max depth, and accordingly will have different tree size. So to OO design the project, the fitness associated with the individual will be calculated, and the size of the tree will be calculated as well. Potentially later on when we swap subtrees and mutations, once we have done changes on individuals, we will need to update the individual fitness and size as well. Just like above avgFitness function, loop through the individuals within the population, sum the sizes, and finally return the floating value of sum divided by the population size.

3 Results

The simply project works pretty well with all the codes Dr. Soule has handed to us. I generated a population of size 100, print the population values out as listed. I have printed out the minimum fitness in the population and the average fitness as well.

```
jenny@jenny-G50VT ~/docu/572/a $ mc
\rm *.o *~ a.out
rm: cannot remove ‘’*~: No such file or directory
make: *** [clean] Error 1
jenny@jenny-G50VT ~/docu/572/a $ m
g++ -c population.cpp
g++ -c individual.cpp
g++ -c node.cpp
g++ test.cpp population.o individual.o node.o
jenny@jenny-G50VT ~/docu/572/a $ a
-1
        /3
                -1
                        -1
                                X4
                                -1.908
                        *2
                                9.65
                                X4
                *2
                        /3
                                14.216
                                X4
                        -1
                                5.918
                                -4.246
        /3
                *2
                        *2
                                3.614
                                X4
                        +0
                                -2.356
                                X4
                +0
                        -1
                                X4
                                X4
                        *2
                                11.904
                                X4
1,0.365015,1 , 2,-0.104971,4 , 3,-0.694687,9 , 4,-1.40413,16 , 5,-2.23331,25 ,
Fitness = 33.997
i Size = 31


Individual j copied from i:
-1
        /3
                -1
                        -1
                                X4
                                -1.908
                        *2
                                9.65
                                X4
                *2
                        /3
                                14.216
                                X4
                        -1
                                5.918
                                -4.246
        /3
```

```
                *2
                        *2
                                3.614
                                X4
                        +0
                                -2.356
                                X4
                +0
                        -1
                                X4
                                X4
                        *2
                                11.904
                                X4
```

```
1,0.365015,1 , 2,-0.104971,4 , 3,-0.694687,9 , 4,-1.40413,16 , 5,-2.23331,25 ,
Fitness = 33.997
1,58.4476,1 , 2,77.9149,4 , 3,53.5319,9 , 4,-18.1335,16 , 5,-139.57,25 ,
Fitness = 197.472
1,-165.731,1 , 2,-165.742,4 , 3,-165.745,9 , 4,-165.747,16 , 5,-165.748,25 ,
Fitness = 395.682
1,0.052183,1 , 2,1.5073,4 , 3,6.659,9 , 4,14.2349,16 , 5,23.9804,25 ,
Fitness = 4.09239
1,-0.0156754,1 , 2,-0.0118131,4 , 3,-0.0139108,9 , 4,-0.0561052,16 , 5,0.0129643,25 ,
Fitness = 31.3134
1,1.84144,1 , 2,1.1142,4 , 3,0.871787,9 , 4,0.750581,16 , 5,0.677857,25 ,
Fitness = 29.9869
1,-2.66008,1 , 2,-13.9558,4 , 3,-25.352,9 , 4,-36.7734,16 , 5,-48.2048,25 ,
Fitness = 98.2845
1,-2.23128,1 , 2,-2.98609,4 , 3,-2.9696,9 , 4,-2.81827,16 , 5,-2.65537,25 ,
Fitness = 36.352
1,1436.09,1 , 2,658.644,4 , 3,376.662,9 , 4,218.545,16 , 5,109.975,25 ,
Fitness = 1634.46
1,-5.67442,1 , 2,-5.45306,4 , 3,-5.35365,9 , 4,-5.3109,16 , 5,-5.30775,25 ,
Fitness = 41.3842
1,-128.262,1 , 2,-234.195,4 , 3,-318.087,9 , 4,-379.94,16 , 5,-419.752,25 ,
Fitness = 731.44
1,-4.06564,1 , 2,-27.289,4 , 3,-69.729,9 , 4,-114.935,16 , 5,-134.779,25 ,
Fitness = 223.33
1,9.85366,1 , 2,10.8537,4 , 3,11.8537,9 , 4,12.8537,16 , 5,13.8537,25 ,
Fitness = 16.3598
1,-11.8628,1 , 2,-13.9183,4 , 3,-15.9547,9 , 4,-17.9804,16 , 5,-19.9996,25 ,
Fitness = 65.4896
1,112.76,1 , 2,75.7648,4 , 3,42.9155,9 , 4,12.3645,16 , 5,-16.9716,25 ,
Fitness = 143.407
1,-111.936,1 , 2,-6.47172,4 , 3,-14.5265,9 , 4,-36.354,16 , 5,-76.0152,25 ,
Fitness = 162.366
1,-47.5364,1 , 2,10.3902,4 , 3,3.24967,9 , 4,2.12029,16 , 5,1.71152,25 ,
Fitness = 56.2555
1,37.4604,1 , 2,11.4321,4 , 3,5.15814,9 , 4,2.57862,16 , 5,1.2778,25 ,
Fitness = 46.2843
1,-13.1005,1 , 2,-13.3473,4 , 3,-13.7705,9 , 4,-14.2744,16 , 5,-14.8219,25 ,
Fitness = 59.3344
1,-1532.46,1 , 2,-116.265,4 , 3,-20.7304,9 , 4,-4.09699,16 , 5,0.104663,25 ,
Fitness = 1538.79
1,9.592,1 , 2,13.178,4 , 3,14.8907,9 , 4,16.135,16 , 5,17.192,25 ,
Fitness = 15.9293
1,0.96272,1 , 2,-10.3499,4 , 3,-19.6265,9 , 4,-26.8671,16 , 5,-32.0717,25 ,
Fitness = 78.2315
1,-18.7049,1 , 2,-24.5533,4 , 3,-8.0773,9 , 4,57.5127,16 , 5,209.722,25 ,
Fitness = 193.237
1,-99.5199,1 , 2,-16.5708,4 , 3,-3.67187,9 , 4,0.0118593,16 , 5,1.33705,25 ,
Fitness = 107.255
```

1,126.654,1 , 2,128.946,4 , 3,131.239,9 , 4,133.532,16 , 5,135.825,25 ,
Fitness = 269.144
1,-0.0239872,1 , 2,-0.0348031,4 , 3,-0.0488797,9 , 4,-0.0649468,16 , 5,-0.0836736,25 ,
Fitness = 31.4083
1,-8.09289,1 , 2,-15.1954,4 , 3,-159.41,9 , 4,17.3216,16 , 5,7.70814,25 ,
Fitness = 170.628
1,-20.5539,1 , 2,-17.6671,4 , 3,-16.8388,9 , 4,-17.7896,16 , 5,-21.5176,25 ,
Fitness = 70.0521
1,20.7343,1 , 2,35.0807,4 , 3,41.9193,9 , 4,43.5132,16 , 5,42.3376,25 ,
Fitness = 59.1329
1,16.6513,1 , 2,45.1578,4 , 3,85.0172,9 , 4,136.229,16 , 5,198.794,25 ,
Fitness = 228.86
1,-12.2767,1 , 2,-16.608,4 , 3,-19.3893,9 , 4,-21.7843,16 , 5,-24.0263,25 ,
Fitness = 72.3749
1,485.896,1 , 2,1059.84,4 , 3,1713.51,9 , 4,2446.91,16 , 5,3260.04,25 ,
Fitness = 4542.04
1,-63.6543,1 , 2,-63.2024,4 , 3,-65.0098,9 , 4,-68.5786,16 , 5,-74.1572,25 ,
Fitness = 176.521
1,1997.84,1 , 2,2387.21,4 , 3,2767.43,9 , 4,3142.36,16 , 5,3512.77,25 ,
Fitness = 6262.16
1,0.00303891,1 , 2,0.00102794,4 , 3,0.000517799,9 , 4,0.000312074,16 , 5,0.000208673,25
Fitness = 31.2883
1,0.527662,1 , 2,3.00785,4 , 3,10.3516,9 , 4,27.2251,16 , 5,59.9866,25 ,
Fitness = 36.7845
1,-1,1 , 2,-1,4 , 3,-1,9 , 4,-1,16 , 5,-1,25 ,
Fitness = 33.0757
1,-61.4084,1 , 2,-60.1009,4 , 3,-58.7934,9 , 4,-57.4859,16 , 5,-56.1783,25 ,
Fitness = 156.811
1,19.5549,1 , 2,52.0957,4 , 3,53.7498,9 , 4,28.4268,16 , 5,-22.8331,25 ,
Fitness = 84.2761
1,-36.6354,1 , 2,-35.6354,4 , 3,-34.6354,9 , 4,-33.6354,16 , 5,-32.6354,25 ,
Fitness = 103.329
1,94.6401,1 , 2,102.429,4 , 3,105.959,9 , 4,105.231,16 , 5,100.244,25 ,
Fitness = 203.67
1,1126.78,1 , 2,213.347,4 , 3,-12.0766,9 , 4,55.7914,16 , 5,347.639,25 ,
Fitness = 1190.52
1,-1.19706,1 , 2,-2.33993,4 , 3,-3.61014,9 , 4,-5.10271,16 , 5,-7.03863,25 ,
Fitness = 40.937
1,-0,1 , 2,-0,4 , 3,-0,9 , 4,-0,16 , 5,-0,25 ,
Fitness = 31.289
1,56.1359,1 , 2,13.052,4 , 3,-2.6427,9 , 4,-11.49,16 , 5,-17.5984,25 ,
Fitness = 76.3399
1,-0.329509,1 , 2,-0.402685,4 , 3,-0.659089,9 , 4,-0.992966,16 , 5,-1.39238,25 ,
Fitness = 33.1628
1,32.3014,1 , 2,47.0709,4 , 3,61.8389,9 , 4,76.6067,16 , 5,91.3743,25 ,
Fitness = 117.071
1,-130.967,1 , 2,-162.255,4 , 3,-97.0101,9 , 4,64.7675,16 , 5,323.078,25 ,
Fitness = 384.087
1,-1.07061,1 , 2,-0.589772,4 , 3,-0.429493,9 , 4,-0.349353,16 , 5,-0.301269,25 ,
Fitness = 31.9644
1,55.9416,1 , 2,69.3064,4 , 3,95.5593,9 , 4,134.7,16 , 5,186.729,25 ,
Fitness = 234.568
1,-0.0274844,1 , 2,-0.0282611,4 , 3,-0.0308792,9 , 4,-0.0365173,16 , 5,-0.0490574,25 ,
Fitness = 31.3602
1,-20.2564,1 , 2,-22.7898,4 , 3,-25.3231,9 , 4,-27.8565,16 , 5,-30.3899,25 ,
Fitness = 85.6682
1,7307.31,1 , 2,2202.59,4 , 3,1785.4,9 , 4,1630.76,16 , 5,1550.14,25 ,
Fitness = 8142.79
1,-1.46848,1 , 2,-7.32396,4 , 3,-12.4268,9 , 4,-17.3414,16 , 5,-22.1808,25 ,
Fitness = 62.6986
1,3.07804,1 , 2,-0.0766199,4 , 3,-0.0262674,9 , 4,-0.0126946,16 , 5,-0.00711245,25 ,
Fitness = 31.3715
1,-99.0997,1 , 2,-58.3435,4 , 3,-47.6323,9 , 4,-45.9815,16 , 5,-47.5038,25 ,
Fitness = 161.902

1,-0.72585,1 , 2,1.40129,4 , 3,14.1226,9 , 4,42.9235,16 , 5,93.701,25 ,
Fitness = 74.0315
1,-0.8347,1 , 2,-2.13274,4 , 3,-7.1905,9 , 4,11.0011,16 , 5,3.5595,25 ,
Fitness = 28.0677
1,47.8857,1 , 2,18.43,4 , 3,7.74743,9 , 4,2.24417,16 , 5,-1.10963,25 ,
Fitness = 57.2626
1,-0.296716,1 , 2,-4.27706,4 , 3,-38.7106,9 , 4,261.237,16 , 5,122.69,25 ,
Fitness = 268.385
1,109.559,1 , 2,107.788,4 , 3,116.571,9 , 4,132.473,16 , 5,154.805,25 ,
Fitness = 254.055
1,6.13517,1 , 2,3.3706,4 , 3,2.45382,9 , 4,1.99632,16 , 5,1.72211,25 ,
Fitness = 28.418
1,-85.1201,1 , 2,-178.282,4 , 3,-267.118,9 , 4,-354.871,16 , 5,-442.192,25 ,
Fitness = 687.531
1,-5.47967,1 , 2,-2.88242,4 , 3,-2.00035,9 , 4,-1.55946,16 , 5,-1.2975,25 ,
Fitness = 34.7887
1,-68.4779,1 , 2,-4675.98,4 , 3,-1735.29,9 , 4,-2114.27,16 , 5,-2711.81,25 ,
Fitness = 6080.93
1,0.0096819,1 , 2,0.0844439,4 , 3,0.313263,9 , 4,0.824302,16 , 5,1.80914,25 ,
Fitness = 29.3239
1,-0,1 , 2,0,4 , 3,-0,9 , 4,-0,16 , 5,-0,25 ,
Fitness = 31.289
1,-15.4006,1 , 2,-24.4294,4 , 3,-33.4288,9 , 4,-42.4208,16 , 5,-51.4099,25 ,
Fitness = 110.131
1,0.295017,1 , 2,8.65629,4 , 3,23.2696,9 , 4,44.1348,16 , 5,71.2521,25 ,
Fitness = 56.1838
1,-0,1 , 2,-0,4 , 3,-0,9 , 4,0,16 , 5,0,25 ,
Fitness = 31.289
1,41.6893,1 , 2,134.729,4 , 3,293.129,9 , 4,530.9,16 , 5,862.052,25 ,
Fitness = 1032.11
1,23.5755,1 , 2,52.3248,4 , 3,65.0902,9 , 4,55.8715,16 , 5,18.6689,25 ,
Fitness = 87.2976
1,-176.098,1 , 2,-1335.22,4 , 3,2905.91,9 , 4,1466.06,16 , 5,1267.63,25 ,
Fitness = 3723.41
1,198.362,1 , 2,394.39,4 , 3,567.763,9 , 4,718.48,16 , 5,846.54,25 ,
Fitness = 1293.05
1,53209.7,1 , 2,18804.3,4 , 3,13366.3,9 , 4,10722.4,16 , 5,9055,25 ,
Fitness = 59659
1,7.13592,1 , 2,2.71192,4 , 3,0.28792,9 , 4,-0.13608,16 , 5,1.43992,25 ,
Fitness = 30.5067
1,0.86824,1 , 2,0.21706,4 , 3,0.0964711,9 , 4,0.054265,16 , 5,0.0347296,25 ,
Fitness = 31.163
1,162.526,1 , 2,161.892,4 , 3,161.256,9 , 4,160.619,16 , 5,159.978,25 ,
Fitness = 336.654
1,1.98319,1 , 2,2.95405,4 , 3,3.94416,9 , 4,4.93909,16 , 5,5.93595,25 ,
Fitness = 22.6584
1,0.922463,1 , 2,2.1695,4 , 3,2.88584,9 , 4,2.95895,16 , 5,2.27717,25 ,
Fitness = 26.9655
1,-13.9923,1 , 2,-184.601,4 , 3,-577.911,9 , 4,-1260.01,16 , 5,-2296.98,25 ,
Fitness = 2720.3
1,10.9854,1 , 2,-6.9213,4 , 3,-24.828,9 , 4,-42.7347,16 , 5,-60.6415,25 ,
Fitness = 110.216
1,380.984,1 , 2,1386.06,4 , 3,3418.7,9 , 4,7055.41,16 , 5,13149.7,25 ,
Fitness = 15345.7
1,0.0419199,1 , 2,0.084086,4 , 3,0.129133,9 , 4,0.177788,16 , 5,0.230329,25 ,
Fitness = 30.9649
1,-15.5101,1 , 2,-21.6461,4 , 3,-7.65491,9 , 4,26.6926,16 , 5,-45.8314,25 ,
Fitness = 79.6185
1,-0.00570462,1 , 2,-0.00930898,4 , 3,-0.0112454,9 , 4,-0.0127629,16 , 5,-0.0141113,25 ,
Fitness = 31.3114
1,10.2967,1 , 2,21.2539,4 , 3,32.9529,9 , 4,45.3938,16 , 5,58.5765,25 ,
Fitness = 54.3069
1,-64.932,1 , 2,-236.55,4 , 3,-952.335,9 , 4,-2638.53,16 , 5,-5721.38,25 ,
Fitness = 6407.32

```
1,4.46744,1 , 2,7.46744,4 , 3,10.4674,9 , 4,13.4674,16 , 5,16.4674,25 ,
Fitness = 10.2673
1,0.739797,1 , 2,0.353362,4 , 3,0.225842,9 , 4,0.162889,16 , 5,0.125662,25 ,
Fitness = 30.9822
1,60.486,1 , 2,18.1187,4 , 3,9.92293,9 , 4,5.77005,16 , 5,-0.36568,25 ,
Fitness = 66.9839
1,2.44548,1 , 2,2.72711,4 , 3,3.00794,9 , 4,3.28817,16 , 5,3.56797,25 ,
Fitness = 25.7009
1,-0.423357,1 , 2,2.74441,4 , 3,9.16298,9 , 4,18.8434,16 , 5,31.7922,25 ,
Fitness = 7.60578
1,65.0119,1 , 2,66.6603,4 , 3,68.3087,9 , 4,69.9571,16 , 5,71.6055,25 ,
Fitness = 128.937
1,-10.7845,1 , 2,-10.7699,4 , 3,-10.749,9 , 4,-10.7166,16 , 5,-10.6593,25 ,
Fitness = 52.2725
1,0.23208,1 , 2,0.18606,4 , 3,0.109833,9 , 4,-0.0408721,16 , 5,-0.479462,25 ,
Fitness = 31.6336
1,0.0405953,1 , 2,0.00647806,4 , 3,0.00226026,9 , 4,0.00107354,16 , 5,0.000601693,25 ,
Fitness = 31.2852
1,862.937,1 , 2,1939.18,4 , 3,3220.26,9 , 4,4718.17,16 , 5,6444.92,25 ,
Fitness = 8838.88
1,-72.481,1 , 2,-370.641,4 , 3,-898.11,9 , 4,-1578.89,16 , 5,-2284.82,25 ,
Fitness = 2974.48
1,-0.0706768,1 , 2,-0.00403346,4 , 3,-0.000733034,9 , 4,-0.000214766,16 , 5,-8.19048e-05
Fitness = 31.2922
min:4.09239
avg:1401.53
jenny@jenny-G50VT ~/docu/572/a $
```

To indicate the I can copy individuals, I have pased the copying results listed below as well as seen from above results.

Test codes that I have used to generate above results are pasted as well.

```
#include<iostream>
#include<cmath>
#include<ctime>
#include<cstdlib>
#include "node.h"
#include "individual.h"
#include "population.h"

using namespace std;

// g++ -g test.cpp individual.cpp population.cpp node.cpp

double X;

int main(){
        srand(time(NULL));
        Individual i;
    int maxDepth = 4;

    i.generate(maxDepth);
    i.print(i.getNodePtr(), 0);

    i.evaluate_print();
    i.calc_size();
    cout << "i␣Size␣=␣" << i.getSize() << endl;

    cout << endl << endl << "Individual␣j␣copied␣from␣i:␣" << endl;
    Individual j = i.copy();
    j.print(j.getNodePtr(), 0);

    Individual* indi = &i;
    int popuSize = 100;
    Population* popu = new Population(popuSize, maxDepth, indi);
```

```
    cout << endl << endl;
    popu->evaluate_print();
    cout << "min:" << popu->minFitness() << endl;
    cout << "avg:" << popu->avgFitness() << endl;
}
```

4 Conclusions

In order be able to do genetic programming, we need certain data structures that would allow us be able to swap the evoluationary algorithms data in the middle functionally as if we have swapped programs. Like this project, we used the tree structure. As far as we understand the Genetic Programming theory and C++ pointer, the project turned out to be not that hard. And so far, it works pretty well.