Cartesian Genetic Programming (CGP)

Assignment 5 CS472 F19 DUE: Thu Dec 5, 2019 at 5pm PT

REWARD: 250 points

TEST DATA: see class web page

1 The problem

Let's evolve logical expressions of AND, OR, XOR, and NOT. A truth table is given of example inputs the outputs that should result. The truth table need not specify all possible inputs just the subset that counts toward the fitness. Combinations of input values that are not given may optionally have any output. Your solution is fairly constrained in this problem. You must you Cartesian Genetic Programming with the parameters that are passed in with the problem. Your evolutionary parameters are your choice.

Your program will be named cgp. It takes no arguments on the command line. It will read a file in this format from the standard input:

```
 <numinputs>  <numoutputs> <width> <length> <arity>
 <numSamples>
 <example 1>
 <example 2>
     .
     .
     .
 <example <numSamples>>
```

Here is an example of a 3 bit increment function:

```
3 3 3 2 2
7
0 0 0 0 0 1
0 0 1 0 1 0
0 1 0 0 1 1
0 1 1 1 0 0
1 0 0 1 0 1
1 0 1 1 1 0
1 1 0 1 1 1
```

Here is a possible output:

```
FINAL Gen 17 :

0 U INPUT     1 U INPUT     2 U INPUT
3 F  AND (  1  2 )    4 ?  XOR (  1  0 )    5 T  XOR (  1  2 )
6 ?  NOT (  4  1 )    7 T   OR (  3  0 )    8 T  NOT (  2  2 )
9 T  OUT (  7  1 )   10 T  OUT (  5  4 )   11 T  OUT (  8  3 )
Fitness: 21.000000
0 ((( B  and  C ) or  A ))
1 (( B  xor  C ))
2 (( not  C ))
```

The listing reads off the values of the 3 inputs, the 9 internal operator nodes, and the 3 output nodes. For input nodes it shows the node number, whether the input was used or not (indicated by 'U' or 'x') and the word INPUT. For internal operator nodes it is the node number, the last value at that node in an evaluation or '?' if that node is not evaluated, the name of the operator, the links to previous nodes. For output operators it is the node number, the value from the last evaluation, the operator OUT, and the two input links of which only the first one matters.

Then the fitness is printed. This is the number of outputs that were right across all outputs and all examples.

Finally you can use the same technique use a recursive technique to print the logical expression for each output.

Hint: the test problem `equal4.in` is pretty tricky.

Info on the test files:

adder4.in	A 2 bit adder with carry
addernoise4.in	Part of a 2 bit adder with carry
and3.in	Just the and function
bc3.in	The number of one bits given in binary
cube.in	Indicate the things that are 1, 2, and 3 bits away from 0
equal4.in	Is true if the number of 1 bits is the same as the number of 0 bits (tricky)
expression3.in	Just a random truth table
inc3.in	The increment function
majority3.in	When are there more 1s that 0s
multiplex2.in	Binary input tells which of 4 lines to activate
nand3.in	Simple nand
parity2.in	2 bit parity
parity3.in	3 bit parity
rotate4.in	4 bit rotate

2 Submission

Homework will be submitted as an uncompressed tar file to the homework submission page linked from the main class page. You can submit as many times as you like. The LAST file you submit BEFORE the deadline will be the one graded. For all submissions you will receive email giving you some automated feedback on the unpacking and compiling and running of code and possibly some other things that can be autotested. I will read the results of the runs you submit and may read your code.

Have fun.