Computational Intelligence Homework Assignment

For this assignment you will need the LiveCD (or Scheme plus the stack computer code). Let me know if the LiveCD doesn't work for you and we will figure something out).

Overview

The supplied program takes numeric tuples x and f(x) [1] and evolves an 'equation' (f(x)) which matches the input data. This is known as symbolic regression. The 'equation' it evolves is actually in the form of a stack program, similar the the old HP calculators. It's essential just a procedure, using arithmetic operations on an input value (x) which generates an output value (f(x)), just like an ordinary function.

So,

$f(x) = 2x$

would be the stack program,

```
push 2
push x
times
```

I want you to run the program and then make modifications to it, see what happens, and write up your results.

Running the Supplied Program

- First boot from the provided LiveCD. The password is 'ucsc'.
- When the desktop comes up, double click on *Stack Comp*.
- When the program comes up select the 'Language/Choose Language' menu and then choose 'Pretty Big' under 'Legacy Languages'.

[1] Like (2,4) indicating that when x=2 f(x)=4. One might guess then that f(x) is 2x or 2+x .. etc.

- Now click 'Run' (upper right corner). A few moments later you will have evolved a computer program (the equivalent of a function) which will match the input data points.

Evaluating the output

A typical output from the program would be,

```
(program ->)
  (#<procedure:s-pop> ())
  ...
  ...
  ...
(indiv-> ((4 4) (9 9) (20 20) (37 37) (60 60) (89 89) (324 324)))
(context-> ((1 4 0) (1 4 1) (1 4 2) (1 4 3) (1 4 4) (1 4 5) (1 4 10)))
(best-fitness-> 0)
```

program See Appendix for description.

best-fitness Zero is best. This indicates the fitness of the displayed individual.

indiv These are pairs of the correct f(x) and the evolved f(x). If they all match the EC function is correctly predicting the f(x). So (5,4) in the second position would indicate that for the second data point the evolved f(x) yields 5, while the correct answer is 4.

context Has the format (¡constant 1¿ ¡constant 2¿ ¡x-value.). The context is used by the program as an environment to execute the evolved program.

Program Structure

I am not expecting you to become experts in Scheme (a Lisp variant). I just want you to modify a few lines of code (usually just constants) to see what happens to the evolutionary process.

Two semicolons are a comment in Scheme.

Here are the data points for the given f(x),

```
;; Regression of
;; f(x) = (3 * x^2) + (2 * x) + 4
  (list
   (list (list 0) 4)
   (list (list 1) 9)
```

```
(list (list 2) 20)
(list (list 3) 37)
(list (list 4) 60)
(list (list 5) 89)
(list (list 10) 324)))
```

So, this says that for x=0, f(x)=4, and for x=1, f(x)=9; etc.

These points were secretly (from the EC program) generated from the given function. The goal of EC is to find the 'function', or in our case a stack computer program.

Suggestions of What to Change

Please don't limit yourself to my suggestions. Often students have surprised me by changing things that I hadn't expected.

Suggestions,

- Population Size. See `define *max-population-size*`.
- Generation Count. See `define *generation-count*`.
- Population Size. See `define *max-population-size*`.
- Constants. See `define problem-constants`.
- Fitness Cases. See `define problem-fitness-cases`.

To evaluate the 'goodness' of your changes simply look at the fitness, or the match/mismatch in the `individual` tuples.

Appendix

Here's a step by step description of a stack program. Note the stack is always propagates 1's from the stack top.

Step	Instruction	Stk-0	Stk-1	Stk-2	Stk 3
0	—	1	1	1	1
1	s-swap	1	1	1	1
2	s-plus	2	1	1	1
3	s-plus	3	1	1	1
4	s-push 2 (x)	x	3	1	1
5	s-times	3x	1	1	1
6	s-swap	1	3x	1	1
7	s-plus	3x+1	1	1	1
8	s-plus	3x+2	1	1	1
9	s-push 2 (x)	x	3x+2	1	1
10	s-times	3x**2 + 2x	1	1	1
11	s-swap	1	3x**2 + 2x	1	1
12	s-minus	3x**2 + 2x - 1	1	1	1
13	s-times	3x**2 + 2x - 1	1	1	1
14	s-push 1 (constant 4)	4	3x**2 + 2x - 1	1	1
15	s-plus	3x**2 + 2x + 3	1	1	1
16	s-plus	3x**2 + 2x + 4	1	1	1