Laboratory 8 – Week 9

## Fitness Functions - the Scales Problem

## 8.1 Introduction

**Firstly, this worksheet *is* one of the worksheets from which your laboratory worksheets portfolio of work will be assessed.**

This laboratory involves developing the fitness function for the *Scales* problem as discussed in the lecture. We are going to write a class that will generate a random solution and evaluate the fitness. This laboratory is **VERY** important since we will be using the code that we write in a number of subsequent laboratories and worksheets.

## 8.2 Exercise 1: Preliminaries

Familiarise yourself with the lecture entitled “8.1 Search Algorithms, Representation, Fitness and Fitness Landscapes”. Pay particular attention to the section on solving the *Scales* problem.

Create a project and associated class called Lab8. Extract the embedded class objects in Appendix A (two classes) and include them into your project. These two classes will form the basis for this exercise sheet. There is a text file called “1000 Primes.txt” which we will use later. Save this file in a convenient place. Examine the Java code to get an overall view of the programs which we are going to work with.

## 8.3 Exercise 2: The CS2004 Class

The CS2004 class contains some useful code for this laboratory and other laboratories. We will also be adding to and expanding on this class in later worksheets. Currently this class contains three static methods detailed as follows:

**static** **public** **int** UI(**int** aa,**int** bb)

Returns a uniformly distributed random integer between aa and bb inclusive.

**static** **public** **double** UR(**double** a,**double** b)

Returns a uniformly distributed random double between a and b inclusive.

**static** **public** ArrayList<Double> ReadNumberFile(String filename)

Reads in a text file and parses all of the numbers in it, and returns an ArrayList of Double. We will use this method in exercise 8.5.

Add the following main method to the Lab8 class.

**public** **static** **void** main(String args[])

{

**for**(**int** i=0;i<10;++i)

{

**int** x = CS2004.*UI*(3, 12);

System.*out*.println(x);

}

}

Running this should produce a display of ten random integers between 3 and 12 inclusive. Test the code with other values. Modify the code so it produces random real numbers between -1 and 1 inclusive. Run each test several times to verify that the list of numbers produced is different each time.

## 8.4 Exercise 3: The ScalesSolution Class

The ScalesSolution class contains a framework for implementing the fitness function and representation for the *Scales* problem. The class contains a String that holds the binary representation for a potential solution to a *Scales* problem. The fitness function applies the solution allocation represented by the string to set of weights passed as a parameter.

The attributes (data fields) and methods are as follows:

**private** String scasol;

This data field stores the representation/solution for the *Scales* problem in question.

**public** ScalesSolution(String s)

This constructor creates a solution from a specified String. If *any* character in the string is not a '0' or '1' then RandomBinaryString method is called with the size parameter set equal to the size of the parameter string that was passed.

**public** ScalesSolution(**int** n)

This constructor creates a random solution of size n. The RandomBinaryString method is called to create this random string.

**private** **static** String RandomBinaryString(**int** n)

This private method creates a random binary string of length n.

Note: this method is not complete and needs writing.

**public** **double** ScalesFitness(ArrayList<Double> weights)

This evaluates the fitness for the specified solution. An array of weights is passed as parameter. The function should return the *Scales* fitness function evaluated on the solution or -1 if there are fewer weights than the size of the solution.

Note: this method is not complete and needs writing.

**public** **void** print()

This displays the solution being represented with no new line.

**public** **void** println()

This displays the solution being represented with a new line.

Implement the following code:

ScalesSolution s = **new** ScalesSolution("10101");

s.println();

This will create a new solution to the *Scales* problem of 5 weights called x with the odd weights on the right hand side of the scales and the even weights on the left hand side.

What do you get if you run the following lines of code?

s = **new** ScalesSolution("10101x");

s.println();

You will get nothing since the RandomBinaryString method has not been completed! When it is working, it should create a random binary string (a solution) of length six (given that the string passed as a parameter is not valid, i.e. not all '0's or '1's).

You will need to complete both the RandomBinaryString and the ScalesFitness methods. The fitness function can be completed using the Pseudo-Code from the lecture. The RandomBinaryString method can be completed as follows:

For each possible place in the string (n of them) you generate CS2004.UI(0,1). This will randomly generate a zero or one. If you get a zero add the character '0' to the end of the string, otherwise add a '1'. You will need to use a For loop.

Create an ArrayList<Double> containing the value 1, 2, 3, 4 and 10 as in the examples in the lecture. Test the fitness function on these examples (the ones from the lecture). Generate a number of random solutions and evaluate the fitness function on them, verify the accuracy manually.

## 8.5 Exercise 4: Reading in Data

The text file “1000 Primes.txt” contains (surprise, surprise) a list of the first 1000 prime numbers (<http://mathworld.wolfram.com/PrimeNumber.html>, <http://oeis.org/A000040>). We will be using these as the weights. The method ReadNumberFile within the class CS2004 can be used to read this file (and others in a similar format) into an ArrayList of Double. Read in the file and verify that this has been done correctly.

For a small number of weights (say, less than ten) generate a number of random solutions to the *Scales* problems using the prime numbers (for example if you are solving it for 8 weights then the first 8 weights will be used by the fitness function) and verify the results manually.

Generate a number (at least 100 each) of random *Scales* solutions for the first 10, 100, 250, 500 and 1000 primes numbers and evaluate (and average) their fitness. Is generating random solutions a good way of solving the *Scales* problem? What sizes could it work for?

## 8.6 Appendix A

The following classes and text files contains the code and data you will need for this exercise sheet.

|  |
| --- |
|  |