University of Wollongong

School of Computing and Information Technology

CSCI464/964 **Computational Intelligence** **Autumn 2019**

Assignment 3 **(Due: 5:30PM, Wednesday 22 May)** **20 marks**

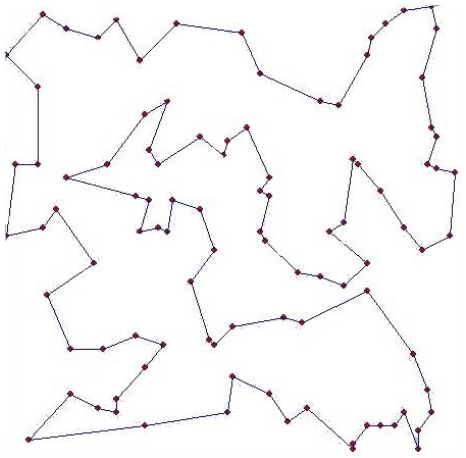
--- Part 1 (Genetic Programming, 10 marks) ---

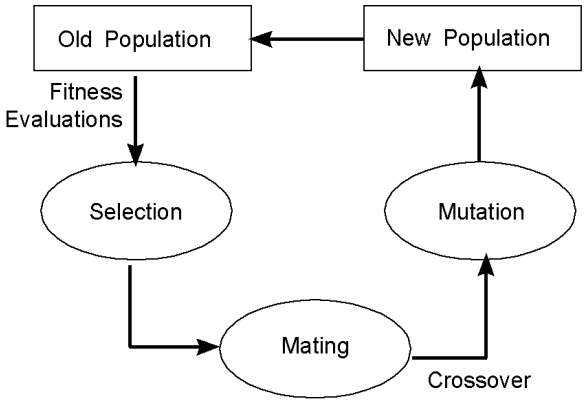
Aim:

This assignment is intended to provide basic experience in solving difficult problems with Genetic Algorithms (GAs). After having completed this assignment you should know how to implement a GA in C++ and find a near optimal solution to hard problems like the Traveling Sales Person Problem (TSP).

Preliminaries:

In the TSP, the goal is to find the shortest distance between N different cities. The path that the sales person takes is called a tour. The following example shows a near optimal tour between 100 cities.





Basic GA Cycle

To test every possible path for an N city tour requires N! math additions. For example, a 30 city tour would be 2.65 X 10^{32} additions. Assuming 1 billion additions per second, this would take over 8,000,000,000,000,000 years. Adding just one more city would cause the number of additions to increase by a factor of 31. Obviously, this is an impossible solution to find mathematically. However, a genetic algorithm (like that shown above) can be used to find a near perfect solution in very little time. The basic concept of Genetic Algorithms is to simulate Darwinian evolution by implementing survival of the fittest. To solve the travelling sales person problem using a GA, first create a group of many random tours in what is called a **population**. These tours are stored as a sequence of numbers representing cities. Second, produce a new population by repeatedly picking 2 of the better (shorter) tours from the population (i.e., parents) and by combining them using the **crossover operator** to create 2 new solutions (i.e., children) in the hope that they create a better solution. The **mutation operator** can also be applied to improve the chance of finding a shorter tour. Crossover is performed by picking a one or two random points in the parents' sequences and switching the numbers (representing cities) in the sequence between the points.

The difficulty in using a GA to solve the TSP is in crossing over the cities of parent solutions to produce child solutions. For example, as show below, crossing over the cities in the parents has resulted in City 1 occurring twice in Child 1 and City 5 occurring twice in Child 2. Furthermore, City 1 is missing in Child 2 and City 5 is missing in Child 1. Consequently, a **more complicated crossover operator** must be used to prevent this occurring.

Parent 1	1 2 3 4 5
Parent 2	3 5 2 1 4
Child 1	1 2 3 1 4
Child 1	3 5 2 4 5

1的45 换2的14

Assignment Specification:

To complete this assignment you are to implement a genetic algorithm for solving the TSP Toll Road problem. The TSP Toll Road problem is the same as the general TSP problem except that the cities have types (1: capital, 2: regional and 3: country) and the cost (cents/km) of traveling between two cities depends on their type. **The objective is to visit all cities at the minimum cost**. The table below shows the cost of traveling between different types of cities:

	1	2	3
1	10	7.5	5
2	7.5	5	2.5
3	5	2.5	1

For example, to travel between a regional city (type-2) and a capital city (type-1) that are 100km apart it would cost 100km x 7.5c = $7.50.

Data: Three data files are provided containing 100, 200 and 500 data items. Each data item represents the coordinates (x, y) of each city (in kms) and the city type (1: capital, 2: regional, and 3: country).

To assist with this task, **a bare-bones GA written in C++ is provide in the file "ga.cpp"**. However, this GA is for solving problems represented with bit strings and **will need considerable modifications and enhancements** to make it capable of finding solutions to the TSP. Your completed program should accept the filename of the TSP data file as a command line argument. If no filename is given your program should ask the user to enter the filename via the keyboard. The output of your program should show the minimum tour distance achieved by each generation. However, if this produces more than two pages of output, in your final program, modify your code so that every 5th, 10th or 20th generation is displayed. **To receive full marks for this assignment the following steps must be completed. (Note: these steps do not have to be implemented in the order given. Step 1 is worth 4 marks. Step 2 is worth 1 mark, Step 3 is worth 2 marks and Step 4 is worth 3 marks.)**

Step 1: To implement the TSP on the code in the file **ga.cpp**, begin by writing a function to read a data file containing the locations of the cities and their type into a dynamically allocated array. Then alter the Init(), Crossover(), Mutate() and EvaluateFitness() functions appropriately to handle the TSP Toll Road problem. **You are encouraged to design your own algorithms for these functions to get your GA performing optimally** (see Step 4). However, to help you get started, the following suggestions are offered.

Init() must be modified to load the initial pop members with random cities. Cities are represented with the numbers 0..n-1 where n is the number of cities in the data file. **Make sure no city occurs more than once in each member**. (Note: **the first number in the given data files is the number of cities in the file**. Use this number to appropriately allocate the size of member arrays etc. so your code will run on any of the given data files. City locations are represented by (x,y) coordinates (i.e., integers: 0..999).)

Crossover() can be implemented in a variety of ways. A search of the Internet should reveal some useful examples. **You should make sure that the crossover operation does not result in any offspring containing duplicate cities.** This can be done by locating any duplicate cities and replace each duplicate city with a missing city. (But which missing city?)

Mutate() can be implemented by swapping two or more randomly selected cities in the member. Feel free to experiment with this to obtain increased performance.

EvaluateFitness() should calculate the fitness by adding all the distances between visited cities times their cost to get the tour cost. This will result in the fittest cities having the smallest cost. Thus you may have to modify how the BestMember is obtained and how parent selection is done. (Step 3 has more info on this.)

Step 2: To avoid the large expense of calculating the same distances during each fitness evaluation, **provide an n x n lookup table** of the cost of traveling between cities where n is the number of cities to be visited. Thus, when fitnesses are calculated the cost of traveling between two cities can be looked up from the table more quickly.

Step 3: involves providing your GA with **two parent selection options**. To do this, implement roulette wheel selection and provide a global constant to switch your GA between Tournament or Roulette Wheel selection. The following algorithm is offered as a suggestion for implementing roulette wheel selection:

```
Copy fitnesses into a temp array
Subtract the minimum pop fitness from each fitness to normalise them
Divide each normalised fitness by the normalized maximum pop fitness to get the scaled fitnesses
Generate a random number between 0 and the sum of all the scaled fitnesses
Set TempSum to 0
for i=0;i<PopSize;i++){
      TempSum += Fitness[i]
      if(TempSum>RandomNumber) return i;
}
```

This should work, however **better performance may be achieved** by scaling the fittnesses such that the fittest member occupies no more than 2 times as much space on the roulette wheel as members with average fitness.

Step 4: Run your GA on all three data sets and experiment with different parameters (e.g., crossover & mutation types and mutation & crossover probabilities) so that your GA finds the cheapest path in the minimum number of generations. Write up the results in your report.

Step 5: Now modify your crossover and mutation operators so that the amount of change they cause to individuals becomes less as evolution progresses. Describe the modifications you have chosen to make in your report. **Provide a brief comment block at the bottom of your program** to explain the measures you have taken and the parameters you have chosen to achieve your result. Submit your code with your optimal parameters in place so your optimal solution can be demonstrated. Make sure the output from your program does not occupy more than one or two pages. (Note: it is ok to just print the fitness of every 5th or 10th or 20th generation.)

mutation 的作用越小

The above is to be taken as a guide only. It is advised to do research on GAs and the TSP problem and make any changes you think fit to improve the performance of the GA on the TSP problem. Please **provide references** if you use techniques found via your research. **Your report should contain sufficient description, discussion and analysis**, in particular,

(1) **Details** of each step (above) of your implementation and any improvements.
(2) The settings, results and performance (**graphs**) of your GA based on your experiments.
(3) Attach a printout of your code. Ensure it is neat, well commented and easy to understand.

--- Part 2 (Deep Learning, 10 marks) ---

Aim: This assignment is intended to provide basic experience in using the recently released open source software library **TensorFlow** developed by Google Brain Team to implement and design simple neural networks for image classification. After having completed this assignment you should know how to realize a linear and convolutional neural network with TensorFlow, understand its training process, and interpret the classification result.

Assignment Specification:

1. Install TensorFlow by following the instruction at https://www.tensorflow.org/install/. Download MNIST data from http://yann.lecun.com/exdb/mnist/. It will be used in the following tasks 2 and 3.
2. **(2 marks)** With TensorFlow, develop and train **a multinomial logistic regression model** with the training set images and labels of MNIST. Then, test your model with the test set images and labels of MNIST. Report the classification accuracy that you can obtain. Describe what this network does and the key steps (such as defining the networks, training, and test) with your own words. Vary the parameters like the number of training data, the learning rate, and the batch size to observe the classification accuracy. Describe your observation.
3. **(3 marks)** With TensorFlow, develop and train **a basic convolutional neural network** with the training set images and labels of MNIST. Then, test your network with the test set images and labels of MNIST. Report the classification accuracy that you can obtain. Describe what this network does and the key steps (such as defining the networks, training, and test) with your own words. Vary the parameters like the patch size and the number of features to observe the classification accuracy. Describe your observation.
4. **(5 marks)** The following images are from the international competition on cell image classification hosted by International Conference on Pattern Recognition in 2014. The images have been pre-partitioned into training set (8701 images), validation set (2175 images), and test set (2720 images), which are provided with this assignment instruction. In addition, a .csv file is enclosed. It contains the category of each image. This file consists of two columns: the first column is the image IDs of all the 13,596 images, and the IDs are consistent with the names of the images in the three sets; and the second column is the category of the cell image. **This is an open task**. You are expected to use the knowledge learned from this subject to achieve the classification accuracy on the test set as high as possible. Note that **you are free to use any classification techniques, toolboxes, software packages, and programming languages to accomplish this task**. For your reference, our research shows that the classification accuracy of 96% is achievable. In addition, a journal paper is enclosed in this assignment for your reference.

5. Write a report on each of the three parts (**parts 1 and 2 shall be put into one single report**). It shall include
   1) A brief introduction on the MNIST data set used in part 2;
   2) As indicated in the **point 2 above**, report the classification accuracy that you can obtain, and describe what this network does and the key steps (such as defining the networks, training, and test) with your own words. Vary the parameters like the number of training example, the learning rate, and the batch size to observe the classification accuracy. Describe your observation;
   3) As indicated in the **point 3 above,** report the classification accuracy that you can obtain, and describe what this network does and the key steps (such as defining the networks, training, and test) with your own words. Vary the parameters like the patch size and the number of features to observe the classification accuracy. Describe your observation;
   4) For the open task in **point 4 above**:
      a. Describe the classification technique, the toolboxes and the language you use;
      b. Describe and plot the structure of the classification system with a diagram;
      c. Describe how you process and prepare the image data when applicable;
      d. Describe how you chose the parameter of this classification system;
      e. Describe the training and validation accuracy and plot them when applicable;
      f. Report the final classification accuracy on the test set;
      g. Elaborate your observation during the course and provide detailed analysis.

Submit:

Submit your program on UNIX via the submit command **before the deadline** and hand in your report with a cover page to in the lecture.

For part 1: C/C++ code that is compliable and runnable on the UNIX platform; Before submitting your code check the format to ensure the format and newlines appear correct on UNIX. (Marks will be deducted

for untidy or incorrectly formatted work.) To avoid formatting problems avoid using tabs and use 4 spaces instead of tab to indent you code. Make sure your file is named: **tsp.cpp**.
For part 2: Provide all the source code of your programs for this part (including points 2, 3 and 4). Since you may use programming languages other than C/C++ and other libraries or packages, **it is not required** that your code be runnable on the UNIX platform. **However, you may be asked to demonstrate** your program on your laptop or a computer that has all the needed environments.

Put both part 1 and part 2 into a single report. Do not write two separated reports.

Submit using the submit facility on UNIX ie:

$ submit -u login -c CSCI964 -a 3 assignment3.zip

where 'login' is your UNIX login ID.

We will attempt to run your program of part 1 on banshee. If problems are encountered running your program, you may be required to demonstrate your program to the coordinator at a prearranged time. If a request for a demonstration is made and no demonstration is done, **a penalty of 2 marks (minimum)** will be applied. **Marks will be awarded for a comprehensive report, correct program design, implementation, style and performance.** Any request for an extension of the submission deadline must be made by applying for academic consideration before the submission deadline. Supporting documentation must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the mark awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than one week late.