**CSC 330: Artificial Intelligence**

Programming Assignment #3 – Genetic Algorithms

# Due Tuesday, March 4, at the beginning of class

***Description:***

The Traveling Salesman problem (TSP) is a famous problem in Computer Science, both because it is a very hard problem to solve (one of a class called *NP-complete*), and because there are many approximation algorithms to find a decent solution to it. TSP can be quickly summarized by assuming that there are *N* cities, with a cost of travel from every city to every other city defined. TSP asks us to find the lowest cost path from a given city back to that city that passes through every other city once and only once.

One way of obtaining a solution to a TSP problem is to use a genetic algorithm. We will be implementing a genetic algorithm to find a solution to TSP problems. This assignment will be more open-ended than assignments you may have previously had.

***Details***:

You may work on this project in pairs, if you wish. Reply to the forum thread stating if you are working with a partner, and if so, who. If you are working alone, post that as well.

Your project should solve any TSP given to it, with a few assumptions. The information for the TSP will be found in a file in the following format. The first line of the file contains a number *N* indicating the total number of cities. After that are *N* lines, each containing *N* numbers. These numbers indicate the cost of traveling from city *i* to city *j*, where *i* is the row and *j* is the number in that row. You may assume that the city graph is completely connected (you can get from any city to any other city). Here is an example of what a file might look like:

5

0 3 1 4 2

4 0 6 1 4

1 2 0 6 5

6 3 1 0 8

4 1 9 3 0

There is a file **tsp.txt** on the Moodle site containing a sample TSP (in fact, it is the above file – the shortest path is 6, for your information). Feel free to write your own test files as well, and also feel free to share them with the class. In fact, I strongly suggest writing your own more substantial test files – the **tsp.txt** file is not very big, and may not be an accurate measurement of your program’s abilities.

Your program should do the following:

* Prompt the user to enter a file name.
* Open the file and read in the information about the TSP. If the file doesn’t exist, you should print an appropriate error message and exit the program.
* Run a genetic algorithm to find the best path for the TSP. **While running, your program should print the cost of the best tour it has found during each generation.**
* When your program is finished, print the best path found, and the resulting path cost.

You are free to store the TSP information any way you like, and to set up the genetic algorithm any way you like. It must be a genetic algorithm, though – using a population of candidate solutions, performing selection, cross-over, and mutation, etc. But the parameters and set-up are totally open. Your algorithm may not find the best solution (in fact, for larger problems it almost certainly will not), but it should “make progress” towards a solution.

This is not a simple assignment, and I strongly suggest that you do not procrastinate. I am not breaking the project into stages, but if I were, I would suggest you accomplish this portion first:

* Successfully read the TSP file and create a set of initially random solutions, and report the best cost tour in that set.

Once this portion is working properly, continue on and implement selection, cross-over, and mutation.

***Extra Credit***:

After all your final projects are submitted, each will be run on a (large and difficult) TSP of my choosing. The program that finds the best solution will receive 3 extra credit points, and the program with the second best solution will get 2 points, and the program with the third best solution will get 1 point. Keep in mind that genetic algorithms are inherently random, so it is possible that a great approach will not find the best answer – it’s just the luck of the draw to a certain extent. Of course, a good approach will give you a better probability, so do some experimentation.