CNSCC.361 Artificial Intelligence Coursework

Introduction:

Marking Scheme

15% of the mark for the CNSCC.361 module is based on the coursework.

At the end of this document, there is an Appendix providing suggestions for a well written report.

Submission

Put your codes in separate folders ("cw_task1" for task1 and "cw_task2" for task2) and call the overall zipped file "cw_lastname_firstname.zip" (replace lastname and firstname with your names). Submit your "cw_lastname_firstname.zip" file and your report on Moodle.

The length of the report should not exceed 5 pages (the format is specified at the end of this document – two column, minimum font size 10pt). It is important to note that we do not want separate reports. There has to be one report.

Task 1

This task of the assignment requires you to perform pre-processing of the real climate data set (temperature and wind speed) measured in Manchester, UK for the period 2010–2015 provided in the file "ClimateData.csv". This data is a subset of publically available (from http://www.worldweatheronline.com) data about climate at Manchester which contain 938 records and five features (i.e., five dimensional vectors) of data from the Summer and the Winter seasons of the period from 2010 to 2014. The meaning of each column of data is listed below:

Temperature, ^{o}C	Wind speed, mph	Wind direction, deg	Precipitation, mm	Humidity, %

Recall from the Lectures and the Lab sessions, the main pre-processing steps:

i) normalization,
ii) standardization,
iii) anomaly detection.

Explain clearly the work of the algorithms, analyze their advantages and disadvantages, provide the code that you developed (do not use downloaded code from elsewhere and be aware about the plagiarism policy of the University) and the results.

From the literature you may find other pre-processing algorithms (e.g. recursive density estimation, PCA, etc.) which you can also mention in your analysis and/or use. For these additional (optional) algorithms you can use available code assuming you correctly make a reference to it; however, demonstrating the understanding of it is necessary. These additional/optional algorithms are for distinguishing between good, average and excellent reports.

Task 2

The *Traveling Salesman Problem* (*TSP*) is one of the most famous problems in computer science. Here we describe the problem and you will implement a Genetic Algorithm (GA) to find a solution, and show and analyse your results. These are to be done in MATLAB. GA has been introduced and discussed as part of a lecture. There was also a lab about GA to give you an initial understanding of the GA approach, but this Task will be applying GA to a different problem than the one in the lab.

TSP consists of attempting to find the shortest complete tour through a series of points (cities), starting and ending with the same point (see Figure 1). Finding the shortest route that visits a set of locations is an exponentially difficult problem: finding the shortest path for 20 cities is much more than twice as hard as 10 cities. An exhaustive search of all possible paths would be guaranteed to find the shortest, but is computationally intractable for all but small sets of locations. For larger problems, optimization techniques, such as GA, are needed to intelligently search the solution space and find near-optimal solutions.

Mathematically, traveling salesman problem can be represented as a graph, where the locations are the nodes and the edges (or arcs) represent direct routes between the nodes. The weight of each edge is the distance between the nodes. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. The goal is to find the path with the shortest sum of weights. Below, we see a simple five-node graph:

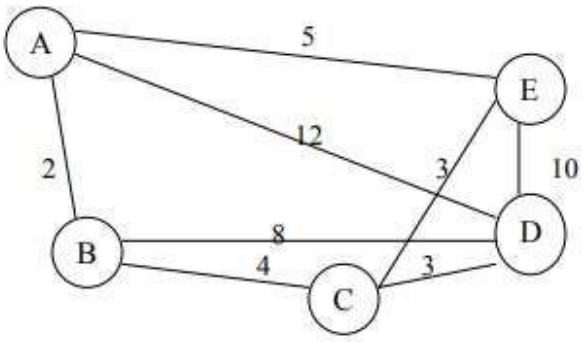


Figure 1- Shortest route example: the problem lies in finding a minimal path passing from all vertices once. For example the path Path1 {A, B, C, D, E, A} and the path Path2 {A, B, C, E, D, A} pass all the vertices but Path1 has a total length of 24 and Path2 has a total length of 31.

In this task, you will be given the (x,y) location of 100 cities in "`xy.mat`" file. So, each population member (chromosome) will have 100 gens.

Finding a solution to the travelling salesman problem requires that you set up a genetic algorithm in a specialized way. For instance, a valid solution need to represent a route where every location is included at least once and only once. If a route contain a single location more than once, or missed a location out completely it would not be valid. To ensure the genetic algorithm does indeed meet this requirement special types of mutation and crossover methods are needed. Firstly, the mutation method should only be capable of shuffling the route, it shouldn't ever add or remove a location from the route, and otherwise it would risk creating an invalid solution. **Question:** What type of mutation? For each selected population member, try three different mutation operators (Swap, Flip and Slide) to generate three new population members. With swap mutation two location in the route are selected at random then their positions are simply swapped. For example, if we apply swap mutation to the following list, [1,2,3,4,5,6,7,8,9] we might end up with, [1,2,5,4,3,6,7,8,9]. Here, positions 3 and 5 were switched creating a new list with exactly the same values, just a different order. Because swap mutation is only swapping pre-existing values, it will never create a list which has missing or duplicate values when compared to the original, and that's exactly what we want for the traveling salesman problem. With Flip mutation two locations in the route are selected at random, and then, the positions between two locations are simply flipped. For example, given two randomly selected locations 3 and 7, if we apply swap mutation to the following list [1,2,3,4,5,6,7,8,9], we end up with [1,2,7,6,5,4,3,8,9]. Moreover, if we apply slide mutation to the list [1,2,3,4,5,6,7,8,9], we end up with [1,2,4,5,6,7,3,8,9]. You also need to pick a crossover method which can enforce the same constraint. What type of crossover? Ordered crossover.

Implement Genetic Algorithm

Your main task is to implement with MATLAB a genetic algorithm that attempts to find a near-optimal solution. You cannot use MATLAB's "ga" function, so you have to implement something similar to what you did in the lab.

Your algorithm should make use of crossover and mutation as described above. Begin with an initial population of at least 50 members and then increase to 200 members (start with 50 members, then try 100, 150 and 200 members). Run your algorithm for at least 1000 generations/iterations and then increase to 10000 (start with 1000 generations, then try 2000, 4000, 6000, 8000, and 10000 generations/iterations). Choose the best ones.

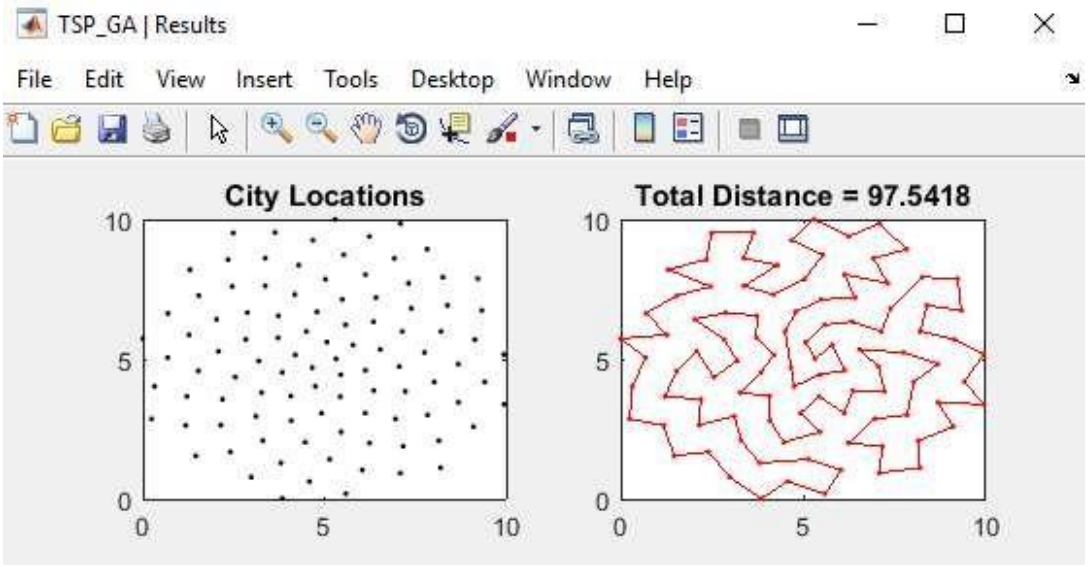
You will need to make many design decisions on how to implement the algorithm and what parameter values to use. For example, you could try different selection methods including roulette-wheel selection, ranking selection and tournament selection to see which one is better. Submit the best algorithm. Your mark will depend not only on the code that you write but also on how well you document your design decisions. In your report, you should also answer the following questions:

What was the fitness score of the most-fit individual in the first generation? What was the fitness score of the most-fit individual in the last generation? Plot the fitness score of the most-fit individual in each generation.

What path did the most-fit individual in the final generation take through the cities? Run the following code to visualize the path of the most-fit individual in the last generation.

```
figure('Name','TSP_GA | Results','Numbertitle','off');
subplot(2,2,1);
pclr = ~get(0,'DefaultAxesColor');
plot(xy(:,1),xy(:,2),'.','Color',pclr);
title('City  Locations'); subplot(2,2,2);
rte = optRoute([1:100 1]);
plot(xy(rte,1),xy(rte,2),'r.-');
title(sprintf('Total Distance =  %1.4f',minDist));
```

Note that "`xy`" variable is a 100 × 2 matrix consisting of the (x,y) location of 100 cities and `optRoute` variable (integer array) is the best route found by the algorithm (i.e., the most-fit individual in the final generation). `optRoute` is 1 × 100 vector. This code will show a figure as shown below but the connections between cities and total distance might be different.



What was the string of 100 digits of the most-fit individual in the final generation?

Run the algorithm 10 times. Does the fitness score of the most-fit individual in the last generation change? If so, why?

Run the algorithm using tournament selection, without cross-over operator and using all three mutation operators (swap, flip and slide) with population of 100 members. Run your algorithm for 10000 generations/iterations. What was the fitness score of the most-fit individual in the last generation? Run the above mentioned code to visualize the path of the most-fit individual in the last generation.

The coursework will be marked based on:

- **Code efficiency**
- **Code commenting and writing style**
- **Presentation and writing of the report**
- **Critical Understanding**
- **Research and Results**
- **Use of Literature**
- **Conclusion and Analysis**

Appendix

Requirements for a Well Written Report

The report should contain:

1. **Title,** name, student number, course, etc., followed by an abstract.
2. **Main part**: Introduction, review of the state of the art. The description of the algorithm and how it performs, including showing results with images. For instance: "This report describes development and application of the k-means clustering algorithm to image processing data…" Give the software code that you used to obtain the results in an Appendix. A very important part of your report is the analysis of the results. For instance, what are the advantages and limitations of the algorithms that you used? How can you characterize the results? Are they accurate?)
3. **Conclusions:** should describe briefly what has been done, with a summary of the main results and outline of the possible future work.

The objective of the assignment is to conduct data analysis on a set of data, and present conclusions on the results.