Evolutionary Computation 2015
Practical Assignment:
Evolutionary Algorithms to Solve the Traveling Salesman Problem

Honours Module

Department of Computer Science
University of Cape Town, South Africa

PROBLEM DESCRIPTION

The *Traveling Salesman Problem* (TSP) (Lin, 1965) may be stated as follows: A salesman is required to visit each of *n* given cities once and only once, starting from any city and returning to the original place of departure.

What route, or tour, should be chosen to minimise the total distance traveled?

Mathematically, the problem may be stated as follows:

Given a cost matrix: $D = (d_{ij})$, where d_{ij} = cost of going from city *i* to city *j* $(i, j = 1, 2, \ldots, n)$, find a permutation $P = (i_1, i_2, i_3, \ldots, i_n)$ of the integers from 1 through *n* that minimises the quantity:

$$d_{i1i2} + d_{i2i3} + \ldots + d_{ini1}$$

ASSIGNMENT

Implement an *Evolutionary Algorithm* (EA) to solve the *symmetric TSP* as *efficiently* and *effectively* as possible. That is, finding a minimal tour length in the least number of generations. The EA must use the following parameter constraints:

- **Number of cities n = 1000.**
- **One EA run = 100 generations.**
- **Number of EA runs = 50.**

Thus, the objective is to design an EA that runs for 100 generations *only* and during this time generate the shortest tour length possible. As a guideline, Java application source code framework for solving the TSP has been provided. The framework includes the following:

- Automated output to file of statistics for the *average*, *minimum* and *maximum* tour length yielded by the EA over *N* runs.
- $TravelingSalesman.java$: Implements the TSP user interface and should implement the main loop of your EA (selection and replacement methods).
- $City.java$: Implements methods for defining city positions and distances.
- $Chromosome.java$: Should implement your EA's variation methods (genetic operators).

You are free to modify the given code framework as you see fit when designing your own EA.

However: The fitness and statistics functions and user interface (600 x 600 map size) must not be changed. ***EAs run with different map sizes will not be marked*****.**

Once you have implemented your EA, the application can be run from the command line. For example:

> **TSP 1000 500 50**

Executes the EA on the TSP for 1000 cities, with a population size of 500 and 50 runs.

> **TSP 1000 500 50 gui**

Executes the EA on the TSP for 1000 cities, with a population size of 500 and 50 runs, but with the 600x600 graphical display in a window showing the progress of the EA when solving the TSP. In both cases results (final tour length at the end of each run for all runs) are written to a text file *results.out* in the local directory.

RUNNING EXPERIMENTS AND STATISTICAL TESTS

An integral part of any empirical based research, including evolutionary computation, is the use of statistical tests to determine if there is a statistically significant difference between the average results of comparative data sets.

- Statistically compare the result data-set for your EA (average, maximum and minimum fitness values for the 50 EA runs) with the result data-set of a classmate's EA.
- You can apply a statistical test of your choosing to ascertain if there is a statistically significant difference between the two EA results data-sets. However, be sure to check if the data-sets are *parametric* or *non-parametric* before applying the test for statistical significance.

EVALUATION

The following will be used to calculate the assignment mark.:

- **Total = 0.5 (*Complete README file, ...*) + 0.5 (*EA task performance*).**
- **EA task performance:** The difference between the ***best fitness*** of a ***benchmark EA*** and ***best fitness*** of your EA calculated for 100 generations and 50 runs.
- **For example:** If the *benchmark EA best fitness* $= 213204$, and *your EA best fitness* $= 227201$, this amounts to ≈ 0.06 best fitness difference. Thus, the score for the EA task performance component would be $= 0.47$.
- **Complete README file, source code and class files, and statistical test results:** See HAND-IN section below.

HAND-IN

A ZIP file (use your student number as the file name) must be uploaded to Vula by 1 June, 23.59, 2015. The ZIP file must contain the following:

- All source code and class files for the EA. The code and classes of your EA should be placed in separate directories.
- Results file for your EA runs (50 runs and 100 generations per run) **and** the results file for your classmate's EA runs.
- A text file labeled "README" that contains the following.:

  – Whose EA you compared fitness results with? Be sure to include their name and student number as well as the average, maximum, and minimum fitness scores of their EA.

  – The results of statistical tests that compare the fitness results of your EA with a classmate's EA. You should clearly state if there is a statistically significant difference between the *average*, *maximum* and *minimum fitness* of your EA versus your classmate's EA.

  – A working theory as to why there is (or is not) a statistically significant difference between the fitness results of the two EAs (**200 words maximum**).

ZIP file submission deadline: Monday, 1 June, 23.59, 2015

References

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 10(1):2245–2269.