Evolutionary Computation 2016
Practical Assignment:
Evolutionary Algorithms to Solve the Dynamic Traveling Salesman Problem

Honours Module

Department of Computer Science
University of Cape Town

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:

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

A variation on this original TSP problem, is where the cities are replaced with mobile "trade points". At given time periods, the positions of the trade-points not yet traversed by the salesman, move to a new pre-defined location. This is the *dynamic TSP problem* (Psaraftis, 1988), a dynamic optimisation problem with real world applications in vehicle and network traffic routing. In this instance of the dynamic TSP, trade-points randomly move (to one of five pre-set locations) at fixed time periods.

ASSIGNMENT

Implement an *Evolutionary Algorithm* (EA) to solve the *dynamic TSP* as *effectively* as possible. That is, *minimising tour length* within 100 generations. The EA must use the following parameter values:

- **Number of cities ("trade-points")** n **= 50.**
- **One EA run = 100 generations.**
- **Population size** $\leq$ **100**
- **Number of EA runs = 100.**

As a guideline, Java application source code framework for solving the TSP is provided. The framework includes:

- Automated output of statistics (*results.out*) for *absolute minimum* and *average minimum* tour length over 100 runs.
- $TravelingSalesman.java$: TSP user interface that 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, but ...

Parameters for population size, number of generations, fitness and statistics functions and user interface (600 x 600 map size) must not be changed. ***EAs run with different map sizes or parameter values will not be marked*****.**

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

> **TSP 100**

Executes the EA on the TSP for 100 runs.

> **TSP 100 y**

Executes the EA on the TSP for 100 runs, but with the GUI displayed.

RUNNING EXPERIMENTS AND STATISTICAL TESTS

A key 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 minimum fitness values for the 100 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 significant difference between the two EA results data-sets, but check if data-sets are *parametric* or *non-parametric* before applying a test for statistical significance.

EVALUATION

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

- **Total = 0.3 (*Complete ZIP file*) + 0.7 (*EA task performance*).**
- **EA task performance:** Difference between ***average best fitness*** of a ***benchmark EA*** and your EA's ***average best fitness***.
- **For example:** (Normalised) *benchmark EA average best fitness* $= 1.0$ and *your EA best fitness* $= 0.6$. So, the score for the EA task performance component $= 0.42$.
- **Complete ZIP file containing README file, source and class files and statistical test results**.

HAND-IN

A ZIP file (use your student number as the file name) which must contain:

- 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 **and** 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 best and best fitness scores of their EA.

  – The results of statistical tests that compare the fitness results of your EA with a classmate's EA. You must clearly state if there is a statistically significant difference between the *average best 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**).

Submission deadline: Monday, 23 May, 23.59, 2016

References

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

Psaraftis, H. (1988). Dynamic vehicle routing problems. In Golden, B. and Assad, A., editors, *Vehicle Routing: Methods and Studies*, pages 223–248. Elsevier, Amsterdam, Netherlands.