Simulation Course (ETS061) 2016-2017

Home Assignment 3: Genetic Algorithm

*** The deadline for this homework is June 9, 2017.**

*** The answers must be delivered in a written report.**

*** All MATLAB source files must be delivered.**

You are provided with a basic implementation of a genetic algorithm for Traveling Salesman Problem (TSP) with closed tours. A closed tour starts from a node (i.e. a city) and ends at the same node. You can download the program from the course homepage under the home assignment 3 via link *GA_TSP.zip*. When you unzip it, you will find in the subfolder GA_TSP the source code and the data files needed to run the genetic algorithm. There are three data files with size 48, 70, and 96 cities. You should run the Matlab scripts *loadatt48.m*, *loadst70.m,* and *loadgr96.m* to create the three data files, respectively. To change the parameters and run the GA program you should use the Matlab script *main_ga.m*. The actual implementation of the GA operations are in the script *tsp_ga.m.* This program currently provides the basic structure of a GA including a random generation of initial population, a random mutation operator, and some necessary data structures such as the distance matrix (dmat). The purpose of this home assignment is to complete the provided program by implementing and adding the following GA operations:

a. ***Selection operation:*** Implement a *roulette wheel* selection algorithm (see the last example in Lecture H3), which selects a population with a given size of $popSize$ from the current population. The selection must be performed based on the fitness of the individuals. Fitness of an individual v (i.e. a tour) is defined as $\frac{1}{distance(v)}$ where $distance(v)$ is the total distance of the tour v. Based on this, you should obtain the cumulative probability distribution of the population, and use it for the *roulette wheel* selection.

b. ***Elitism:*** a fraction $eliteFract$ of the best individuals from the current generation (i.e. current population) must directly go to the next generation by replacing the worst individuals in the current population. The selection of the elite individuals must be performed before the *roulette wheel* selection, but they must be transferred to the next population after applying the cross over and mutation operations (the order of operations is clearly specified in the provided code in script *tsp_ga.m*).

c. ***Cross-over operation:*** cross-over is performed with a given probability $crossProb$. This is basically the fraction of the population that should undergo the cross over operations. Make sure an even number of individuals are selected in this step. Each cross-over operation involves two parents chosen purely random from the current population (selection of parents does not depend on their fitness). Then, the two parents are combined by the following procedure to produce exactly one offspring, which is then replace one of its parents chosen at random:

```
Crossover (parent $P_1$ , parent $P_2$)
    K = $\lfloor 0.3 * size(P_1) \rfloor$    ($\lfloor \quad \rfloor$ is floor operator, and $size(P_1)$ is the number of cities in $P_1$)
    $T_l = P_l$;
    $T_2 = P_2$;
    O = ε;                          (O is the offspring, and ε is an empty string)
    While $T_l \neq$ ε do
        Split $T_l$ in two sub-tours $T_{11}$ and $T_{12}$ such that length of $T_{11}$ = min {length of $T_1$, K};
        Append $T_{11}$ to O;
        Update $T_l$ by removing from it the cities in $T_{11}$
        Update $T_2$ by removing from it the cities in $T_{11}$
        X = $T_l$;
        $T_l = T_2$;
        $T_2$ = X;
    End {while};
    Return O as the offspring
```

Mutation operation: you should perform 2-opt mutation with a given probability $mutProb$ on the offspring generated by the cross-over operation. The 2-opt mutation is performed by swapping two randomly chosen cities in a tour. This has already been implemented in the code provided for mutation; however, this code performs mutation on a randomly chosen individual from the population. Instead, you should modify the mutation code so that it is performed on the offspring created by cross-over.

Simulation Tasks: with the above operations implemented and validated, set the simulation parameters as follows: $crossProb = 0.25$, $mutProb = 0.5$, $eliteFract = 0.02$, and $popSize = 200$.

> **Task 1)** set the number of iterations to $numGen = 4000$, and run your simulation once for each of the three data files. At the end of each iteration, record the solution with the best fitness value (i.e. the tour with minimum distance in the current population) and the average fitness of the population. Then, using your recorded values for all iterations, plot a graph for each data file, showing the best and average fitness with respect to iteration number. Using these plots, explain the convergence behavior of GA for the three data files (i.e. do they converge? How quickly they converge?).
>
> **Task 2)** use different values for the number of iteration ($numGen$) from 100 to 2000 with a step size of 100. For each value of $numGen$, run your simulation 15 times and record the average cost (i.e. distance) of the best tour found by GA with a 95% confidence interval for 15 runs. Using these results, plot the average cost (together with the confidence interval) with respect to $numGen$. Do this experiment separately for the three data files *loadatt48*, *loadst70,* and *loadgr96.* By observing your plots, explain the convergence behavior of GA for each data file scenario and with respect to $numGen$. For which data file the confidence interval is tighter and what does this mean?