ANKARA YILDIRIM BEYAZIT UNIVERSITY
CENG 103 TERM PROJECT
FALL 2017

TRAVELING SALESMAN PROBLEM via GENETIC ALGORITHM

Part 1. Biology

Human beings and all other organisms consist of codes in which physical, anatomical and perhaps behavioral features are described according to. These codes and the rules determined by are embedded in the *genes* of organisms. The connected genes together into long strings are called *chromosomes* in where traits of organisms such as eye color or skin color are coded by means of genes. When organisms mate, they combine their genes revealing new organisms with new traits. This process is known as *recombination* and shared genes may sometimes *mutated* in the resultant offspring. As the life progresses in nature, organisms repeat this process over and over until natural selection is completed or optimized. Genetic algorithm is a way to solve some complex problems by adapting the corresponding problem to the built in optimization process in nature which is mentioned above. In other words, the process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be transferred to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found. A simple flowchart for the genetic algorithm is shown in Figure-1.

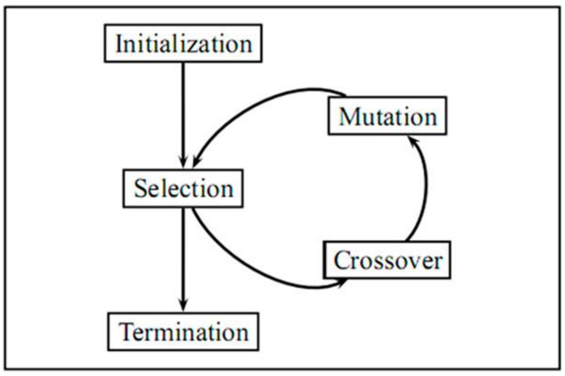


Figure-1 Flowchart for GA

Part 2. Algorithm Steps

The algorithm steps of genetic algorithm may be defined as follows.

- *Initialization* - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
- *Evaluation* - Each member of the population is then evaluated and we calculate a fitness for that individual. The fitness value is calculated by how well it fits with our desired requirements. These requirements could be simple, 'faster algorithms are better', or more complex, 'stronger materials are better but they shouldn't be too heavy'.
- *Selection* - We want to be constantly improving our populations' overall fitness. Selection helps us to do this by discarding the bad designs and only keeping the best individuals in the population. There are a few different selection methods but the basic idea is the same, make it more likely that fitter individuals will be selected for our next generation.
- *Crossover* - During crossover we create new individuals by combining aspects of our selected individuals. We can think of this as mimicking how mating works in nature. The hope is that by combining certain traits from two or more individuals we will create an even 'fitter' offspring which will inherit the best traits from each of its parents.
- *Mutation* - We need to add a little bit randomness into our populations' genetics otherwise every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individual's genome.
- *And repeat!* - Now we have our next generation we can start again from step two until we reach a termination condition.

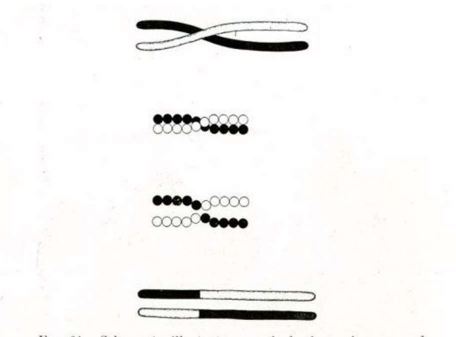


FIG. 64. Scheme to illustrate a method of crossing over of the chromosomes.

Figure-2 An illustration of crossing over of two chromosomes

a. Initialization

The process begins with a set of individuals which is called a population. Each individual is a solution to the problem you want to solve. An individual is characterized by a set of parameters (variables) known as genes. Genes are joined into a string to form a chromosome (solution). In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome. Figure-3 shows an example for gene, chromosome and population concepts.

b. Evaluation

The fitness function for evaluation determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

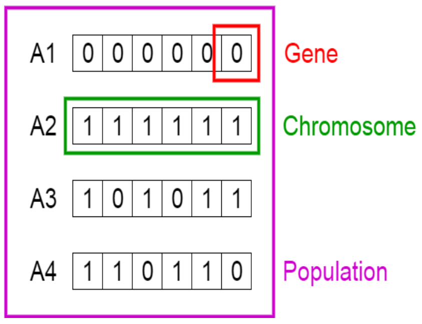


Figure-3 Gene, chromosome and population concepts

c. Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction or recombination.

d. Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. For example, consider the crossover point to be 3 as shown below in Figure-4.

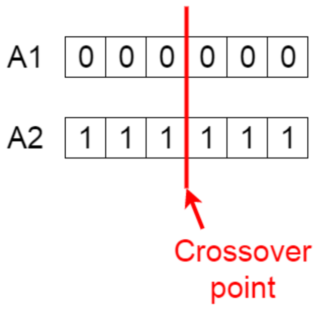


Figure-4 Crossover

Then the offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. You can see Figure-5 for visualization.

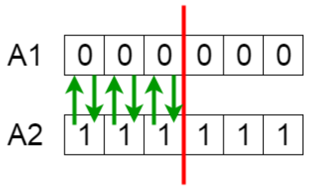


Figure-5 Exchanging genes among parents

In the following step, the new offspring are added to the population as shown in Figure-6.

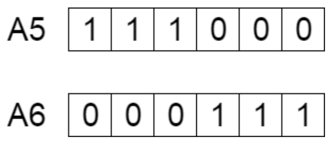


Figure-6 New offspring

e. Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

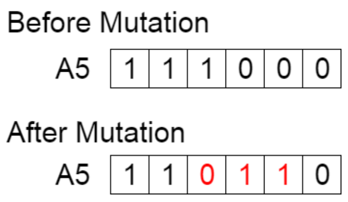


Figure-7 Mutation: Before and After

Mutation occurs to maintain diversity within the population and prevent premature convergence.

f. Termination

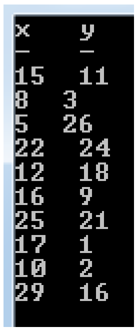
The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

Part 3. Traveling Salesman Problem

We can define traveling salesman problem as follows. Assume a salesman wants to sale his goods in several cities starting from a selected one. However, he must visit each city at most once and at the end he expects to proceed shortest distance by visiting all of the cities available. For this reason, you are expected to write such a C code in which the output declares shortest route for the salesman by taking into account that each city is visited at most once. To do so, this problem requires to be mimicked the natural selection process in nature by exploiting the genetic algorithm.

Part 4. TSP via GA

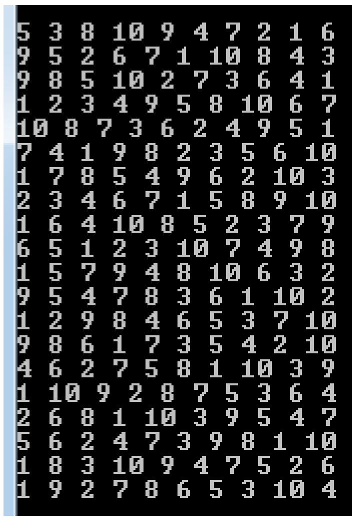
The first thing to solve traveling salesman problem with genetic algorithm is constructing a population. Notice that in this project what we try to find is the shortest route between cities for traveling salesman. For this reason, you are asked to generate N cities (the number of cities will be defined as global variable) with x and y coordinates. Please see Figure-8 as an example of city generation with N is 10 and the coordinate range is [0,30]. These coordinates will be generated randomly by using the function *rand()*. The range of coordinates will be declared as global variables as well. In other words, you are going to write a function which generates N cities whose coordinates vary in a range determined by you. This is the first step of the project. In the following step, you need to find the distances between each city. In other words, a two dimensional array (NxN) will store the distance values of each pair of cities. This is shown in Figure-11. So far, we have a map of cities and the distances between them. Since the aim is to find the shortest route between the cities in this map, we can assume that a population in this problem should consist of random routes. A random route on the other hand may be indicated with the numbers representing each single city as shown in Figure-9.



```
x   y
-   -
15  11
8  3
5  26
22  24
12  18
16  9
25  21
17  1
10  2
29  16
```

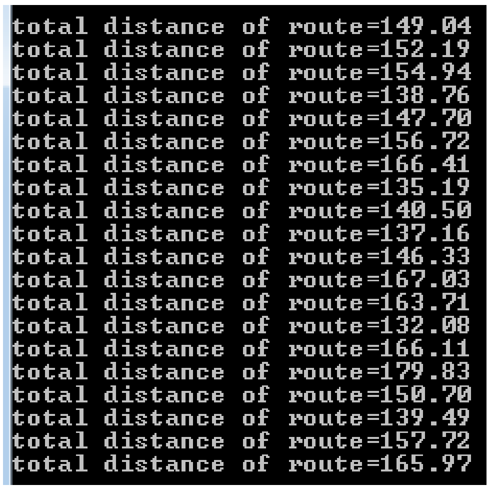
Figure-8 City coordinates

Each row in Figure-9 represents a route for a map in which 10 cities are located. For instance, in the first route, salesman starts with fifth city to sell his goods and travels to third city. After third city, he travels to eighth city and finally he terminates his travel in sixth city. Hence, he completes his travel by visiting each city and at most once (moreover, in this step you need to make sure that each route in the corresponding population is different from each other). However, according to Euclidean distance between these cities, this route may not be the shortest one, requires calculation of total distance of each route to understand the fitness score of each route. You are going to use fitness score of a route in order to see how short a route is to travel. Recall that routes with high score will be selected for crossover or mating step (see Fig.1). For this reason, after generating a population (the number of routes in a population will be a global variable) randomly, you need to write a function that calculates the total distance traveled in each route. The next step is to select parents from this population. As aforementioned, the selected parents must be strong in terms of offering short distances in this problem. Even though there are possible methods to select parents from a population, you are asked to implement tournament method in which ten routes randomly selected and two of these ten routes providing shortest lengths are considered as parents in crossover section. In other words, you need to write a function that sorts out route distances of randomly selected ten routes from all population and selects the shortest two routes. The first recombination section is ready to be performed.



```
5 3 8 10 9 4 7 2 1 6
9 5 2 6 7 1 10 8 4 3
9 8 5 10 2 7 3 6 4 1
1 2 3 4 9 5 8 10 6 7
10 8 7 3 6 2 4 9 5 1
7 4 1 9 8 2 3 5 6 10
1 7 8 5 4 9 6 2 10 3
2 3 4 6 7 1 5 8 9 10
1 6 4 10 8 5 2 3 7 9
6 5 1 2 3 10 7 4 9 8
1 5 7 9 4 8 10 6 3 2
9 5 4 7 8 3 6 1 10 2
1 2 9 8 4 6 5 3 7 10
9 8 6 1 7 3 5 4 2 10
4 6 2 7 5 8 1 10 3 9
1 10 9 2 8 7 5 3 6 4
2 6 8 1 10 3 9 5 4 7
5 6 2 4 7 3 9 8 1 10
1 8 3 10 9 4 7 5 2 6
1 9 2 7 8 6 5 3 10 4
```

Figure-9 Possible roots randomly generated showing the population of Traveling Salesman Problem

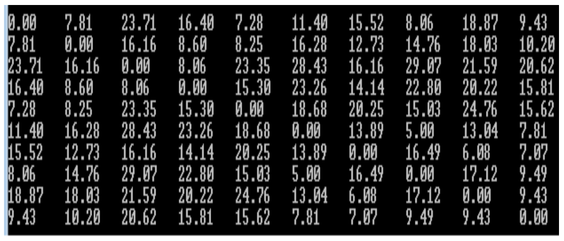


```
total distance of route=149.04
total distance of route=152.19
total distance of route=154.94
total distance of route=138.76
total distance of route=147.70
total distance of route=156.72
total distance of route=166.41
total distance of route=135.19
total distance of route=140.50
total distance of route=137.16
total distance of route=146.33
total distance of route=167.03
total distance of route=163.71
total distance of route=132.08
total distance of route=166.11
total distance of route=179.83
total distance of route=150.70
total distance of route=139.49
total distance of route=157.72
total distance of route=165.97
```

Figure-10 Total distance in each route shown in Fig-8

In the following step, you are going to perform crossover where parents swap their genes from a randomly selected point on chromosome. In other words, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

This procedure called one point crossover and it is depicted in Figure- 12.



```
0.00   7.81   23.71  16.40  7.28   11.40  15.52  8.06   18.87  9.43
7.81   0.00   16.16  8.60   8.25   16.28  12.73  14.76  18.03  10.20
23.71  16.16  0.00   8.06   23.35  28.43  16.16  29.07  21.59  20.62
16.40  8.60   8.06   0.00   15.30  23.26  14.14  22.80  20.22  15.81
7.28   8.25   23.35  15.30  0.00   18.68  20.25  15.03  24.76  15.62
11.40  16.28  28.43  23.26  18.68  0.00   13.89  5.00   13.04  7.81
15.52  12.73  16.16  14.14  20.25  13.89  0.00   16.49  6.08   7.07
8.06   14.76  29.07  22.80  15.03  5.00   16.49  0.00   17.12  9.49
18.87  18.03  21.59  20.22  24.76  13.04  6.08   17.12  0.00   9.43
9.43   10.20  20.62  15.81  15.62  7.81   7.07   9.49   9.43   0.00
```

Figure-11 Distance between each city are stored in a two dimensional array to be used later for overall route length

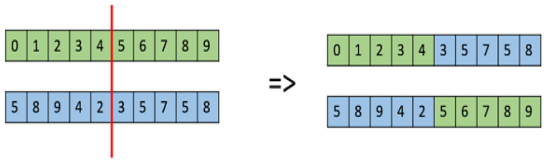


Figure- 12 One-point crossover

Since the recombination reveals childs, the next step is to implement mutation on new individuals to obey the rules of natural process described in genetic algorithm. In simple terms, mutation may be defined as a small random tweak in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability. If the probability is very high, the GA gets reduced to a random search. Mutation is the part of the GA which is related to the “exploration” of the search space of solution. In this project you are asked to implemet swap mutation method in which two positions on the chromosomes are randomly selected and their values are interchanged as shown in Figure- 13.

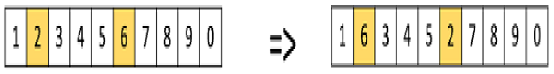


Figure- 13 Swap mutation

Mutation rate in genetic algorithm indicates the probability of genes of chromosomes during the crossover to experience mutation. This rate can take any value between 0 and 1 as expected. During mutation process, hence you need to check whether mutation is required or not according to a predetermined mutation rate as global variable. As a hint, you can perform this step by using *rand()* function. Finally with the mutuation step, recombination is completed and two childs and a new population are revealed. It is worthwhile to note that one of the child and parents with greater fitness scores are replaced in the population and the whole process is repeated over and over until the result remains constant or changing with very small values.

Part. 5 Functions to be written

In this part, to simplify things a little bit more, we will give you a list of functions that you need to declare outside of the *main()* function.

1. A function that produces x and y coordinates for a predetermined coordinate range. You can keep x and y values of cities in one dimensional arrays and you can use rand() function.
2. A function that generates a population which consists of N routes. Each route is an one dimensional array consisting of numbers each representing a city in the route (see any row of Fig.9). As a way, you can create numbers from 1 to N (number of cities) in ascending order and then shuffle each array so that each route differs in terms of the order of cities to be visited.
3. As mentioned before, each route actually is a parent in the population and you need to obtain fitness score for each to be used in selection for the recombination. Since the aim is to obtain shortest route distance in traveling salesman problem, the fitness score will be represented in terms of route distances. Hence, you need to write a function that calculates each route length in the population and stores these values in an array to use later.
4. A function to select two parents according to tournament method described before. This function may also perform crossover by swapping parent chromosomes from a randomly selected point and including mutation if required.
5. The rest is replacing best parent and child to population and repeating whole process again and again until the desired result is obtained. So it is important to build an algorithm that is able to perform iteration over and over.
6. Please feel free to organize your own functions as required for your algorithm. Assume these functions are recommended to implement genetic algorithm, you can derive different functions depending on your own flow chart.
7. You can call your functions in main() function or in other functions. This also depends on your flowchart. Feel free to organize things as you wish.

Part 6. Global variables

Here is a list of variables to be defined as global variable in order for user to change important parameters of the programme confidently.

1. NUMBER_OF_CITY
2. NUMBER_OF_POPULATION
3. COORDINATE_RANGE_END
4. MUTATION_RATE

Part 7. Some Rules

Estimated due date to submit and present your project is the first week of January 2018, the exact date will be announced later. Group study is allowed, each must consist of at most two students. You can also study by yourself. Please send us an e-mail indicating your group members (“only you” or “you and your partner”). Your project report and code will be evaluated in plagiarism detection software automatically and will not be even evaluated (works succesfully) if cheating is detected. You can ask questions regarding your project any time face to face if possible or via e-mail otherwise. Format of the reports will be announced later. Good Luck!

Res. Asst. Çağrı KAPLAN

Res. Asst. Çağrı ASLAN