**Genetic Algorithm based Optimal Route Planning**

Project Report

**Navaneesh Gangala**

***Abstract***

This Thesis Work will concentrate on a very interesting problem, the Vehicle Routing Problem (VRP). In this problem, customers or cities have to be visited and cleaning services have to be provided to each of them, starting from a basis point (Depot) on the map. The goal is to solve the transportation problem, to be able to deliver the services

- on time for the customers,

- using the available resources

- and – of course - to be so effective as it is possible.

Although this problem seems to be very easy to solve with a small number of cities or customers, it is not. In this problem the algorithm has to face with several constraints, for example time constraints should not be less than specified, route length, etc. This makes this problem a so called Multi Constraint Optimization Problem (MCOP). What’s more, this problem is intractable with current amount of computational power which is available for most of us. As the number of customers grow, the calculations to be done grows exponential fast, because all constraints have to be solved for each customers and it should not be forgotten that the goal is to find a solution, what is best enough, before the time for the calculation is up. This problem is introduced in the first chapter: form its basics, the Traveling Salesman Problem, using some theoretical and mathematical background it is shown, why is it so hard to optimize this problem, and although it is so hard, and there is no best algorithm known for huge number of customers, why is it a worth to deal with it. Just think about a huge cleaning service company with ten thousands of vehicles, millions of customers: how much money could be saved if we would know the optimal path for all our services. Although there is no best algorithm is known for this kind of optimization problems, we are trying to give an acceptable solution for it in the second chapter, where Genetic Algorithm is described. This is based on obtaining the processes of nature and material science from the Darwin’s theory of evolution. This algorithm will hardly ever be able to find the best solution for the problem, but it will be able to give a very good solution in special cases within acceptable calculation time.

Finally the possible improvements of these algorithms are discussed.

***Acknowledgement***

This project work could not have been done without the help of my teachers and some of my project mates. I would like to express my thanks especially for Amogh , who gave us the basic idea of this project and showed me the way, how the methods should be used, which I have described in this work. I would also like to express my special thanks to Dr. Sridhar Pappu, Dr. Dakshinamurthy V. Kolluru and all of the Data Scientists in INSOFE. In this programme I learnt a lot of useful methods and techniques in the field of Data Science. The past 5 months – I spent with this course – were very useful for me: I learnt a lot.

1. **The Vehicle Routing Problem (VRP)**

***Basis of the VRP: The Traveling Salesman Problem (TSP)***

The VRP is based on a very simple optimization problem which is called the Traveling Salesman Problem. In this problem there are a given number of cities (n). A Traveling Salesman has to visit all of these (n) cities, each only ones and he should return to the city he was starting his tour from. The goal is to optimize this problem, so to be able to ask the question: “which one is the shortest – or cheapest, depending on the measure used – path for the Salesman?” Solving this problem with a small number of cities seems to be very easy. Increasing the number of the cities, the problem is more complex and complex. This work will not focus on solving the TSP, just as a basis of the VRP, the work will boundary discuss it. This problem was described by Sir William Rowan Hamilton in the 1800s, so it can be said, that TSP was “discovered” in the 1800s. In 1922 – about 100 years later! - Karl Menger published the TSP in its currently known form. Menger published his theory as the “Messenger Problem”: he wanted to find the shortest path for postmen and travelling businessmen. Until that the commonly used solution was choosing the nearest city for the salesman and going there. In the 1940s a few scientists published a study in this topic. The goal was to reduce the mayor costs of carrying man and equipment from one survey to the next in Bengal, through randomly chosen locations. In these years TSP became the “prototype” of a hard problem in combinatorial optimization. During the 1950s several solutions were published for TSP but they were quite far from the optimal one. For example I. Heller published an 88 page paper about a solution he has created for hundreds of cities. In 1956 he described a heuristic method which used the nearest-neighbor algorithm to solve the problem. In 1957 L. L. Barachet published a graphical solution for small number of cities (around 10), that could be used manually and gave more optimal solutions than had been found before. In the 1950s the increasing number of computers made it easier to solve the problem. In these times, several algorithms were published and tested in the available computers, for example on IBM 650. In the 1960s the first solution with dynamic programming was described by R. Bellman. He was able to solve the problem for 17 cities. M. F. Dacey was able to find a heuristic algorithm which was able to solve the problem and the solutions were 4, 8% less effective than the optimal solution, which is a quite impressive result. Later on heuristics were used for 60 and more than 100 cities to solve the problem. In these years papers on branch and bound algorithms were published and implemented for the TSP (J.D.C. Little, K.G. Murty, D.W. Sweeney, and C. Karel on an IBM 7090). In the 1970s M. Held used minimum spanning trees to solve to TSP. Later on integer programming and “branch and cut” algorithms were also implemented.

The problem with the TSP is that increasing the number of cities will increase the size of the solution space exponentially fast.

There are several methods available to solve the TSP, but we will focus on solving Vehicle Routing Problem (VRP) with Genetic Algorithm.

**Brute Force**

The easiest way would be to find all of the possible permutations of the solution space and then selecting the best one. Because of the fast increasing number of the possible solutions, this is not the way to solve this problem, even with having supercomputers.

**Genetic Algorithm**

For solving the TSP, Genetic Algorithm and Simulate Annealing is a good choice. This project Work will not concentrate on solving the TSP, but it will concentrate solving the VRP, which is a more complex – multi constrained (like time should not be exceeded, route length should not be exceeded) – variant of the TSP. The described method in VRP can be easily used also for solving the TSP. This method going to be described later.

1. **The Genetic Algorithm (GA**)

***Introducing GA***

The Genetic Algorithm is a method, based on the example of the natural selection in the “real natural world”. GA uses several algorithms and methodologies to create better and better approximate solutions for a given problem, using the older solutions (parents) to create new ones (children). This will led to an approximate solution of the given solution space. It can be said, that – in the nature – over time it will lead to populations with individuals having better and better abilities of adaptation, and better chances of survivorship.

The GA in the nature: evolutionary biology

Evolutionary biology describes the methodology of adaptation of the different life structures and generations to their environment. To understand the method, some concepts have to be declared.

**Overview**

The population is a given sets of living beings (creatures) in the nature. The population contains organisms from the same kind, each one having different attributes. These attributes are specified by the chromosomes the organism has, and each organism has his own chromosomes. Chromosomes are built up from genes: these genes are responsible for the different attributes the organism has. Based on these genes all representative of the population have different attributes, excepted from the twins.

**Reproduction and selection**

Based on the genes of an issue, it will have more or less chance to live. This will lead to a simple thing: the one with better genes will live, the others will not. This is called “the selection” in the natural genetic algorithm. In computer science it can be modeled in several ways and going to be discussed later. The main goal of the Mother Nature is to select the individuals, based on their skills. These selected individuals are going to be able to reproduce themselves, while the others will not (e.g.: they will not find a partner for mating because of their bad overall fitness.). When the individuals are reproducing themselves, they child individuals will inherit their chromosomes and genes, and with this, the new children individuals will have a new set of chromosomes and with this, a new set of skills. This method contains two main steps:

**Crossover**

This means that the inherited chromosomes are going to be “mixed” by nature from the parent individuals into the children individual(s). This will result a new combination of gene set in the new individuals. Do notice that this does not necessarily means that the better genes will be selected; this only means that one of the parent genes will be selected in the child. This means that there is a possibility, that the child will have worse overall fitness than its parents.

* Example:
  + Parent 1: **X X** | X X X X X
  + Parent 2: Y Y | **Y Y Y Y Y**
  + Offspring 1: **X X Y Y Y Y Y**
  + Offspring 2: Y Y X X X X X

There are two parents are visible (upper side). The X’s & Y’s are the genes of these parents: When mating occurs, crossover will take place: the children will inherit one gene at each position from one parent. In this picture it can be seen, that offspring 1 (bar – vertical line) inherited the first gene from unknown parent, the second gene from the second parent (bar – vertical line), but offspring 2 (bar – vertical line) inherited the first one from unknown parent and the second one from the first parent (bar – vertical line).

**Mutation**

Mutation is a very easily definable method: when mating takes place, some of the genes will not be inherited from the parents, but will be modified randomly. This will led to a new, but unknown result in the child individual, but will diversities the individuals in the following population: for example it is possible, that the mutated gene will be much better than the original ones, and this represents an important skill of the individual (e.g.: ability of fastness). In this case the overall fitness of the individual will be better. For example, it can be imagined as on the picture, child 1 would have a “blue” gene on the 2nd place (this cannot be found in none of the parents).

****

***The GA in Computer Science***

The basic idea of genetic algorithms is coming from the methodology described previously. GA in computer science would like to create an artificial model for the previously written steps to be able to solve problems which could not be solved in the classical way of computing. Before introducing the GA in computer science, the basic methodology and definitions have to be described (like “What is a gene?”, “What is a population?” etc.). Based on this, we will be able to create a “Simulated nature” solving problems which seems to be unsolvable.

Short history of GA

In 1957, Alex Fraser published his method on simulating artificial selection. From this date, computer simulation of evolutionary biology started to become a new field of computer science. In the 1960s, several scientist published methods of genetic selection and the realized the usage of this methodology: they can be used for solving optimization problems with exponential grow in their solution space. In these years the basic definitions – like mutation, crossover – had already been published in computer science. From 1963, when Barricelli reported about a simple simulation “game” he was modeling, artificial evolution become a widespread method solving optimization problems. In the 70’s, scientist were able to solve engineering problems based on evolution strategies. Genetic algorithms become popular in the early 70’s, when John Holland published his book “Adaption in Natural and Artificial Systems”. Although GA was an interesting topic on those days, it remained theoretic until the mid 80’s. In these years, there was an explosion on computational power which allowed GA to become widespread, solving complex optimization processes. GE (General Electric) was the first, who have sold a GA based tool for planning industrial processes. In 1989 the first GA has been sold for desktop computers.

Basis of the GA

The first step is to create an artificial environment for the specified facts of the natural genetic algorithm: “What is a population?”, “What is a gene?”, “How to mutate them?” The goal is to represent all previously described elements in an artificial environment, according to the need of the problem to solve.

**Representing the elements of evolutionary biology**

To be able to create the artificial evolutionary biology, we have to discuss, how natural parts are going to be represented in a genetic algorithm. This chapter will discuss this problem.

**Genes**

This is the basic metrics of the Genetic Algorithm. The gene has a metrics which somehow represents a part of the solution, for example a skill or “goodness” (weight, time, size, etc.) of an element. These genes are going to be inherited, when mating takes place. Genes are basic, mainly numerous elements of the GA. Mainly they are represented in some way of an integer or long, etc.

**Individual, issue, chromosome…**

An individual is a part of the population, containing one possible set of genes. Individuals are mainly different and individuals are defined through their genes. An individual has got an overall fitness function, which is calculated through some mathematical model.

An individual in computer science is one possible solution for the given problem, for example a route in the VRP. In this case the overall fitness is the total length of the route; the genes are the cities, which are visited in the given order (all of these variables might be integers and the individual can be stored in a set of elements, an array). When we build a population, these individuals are compared to each other. The selection and inheritance will also be executed on these individuals. Do notice, that this information have to be stored in a form, so they are comparable to each other, e.g.: some kind of structured array, or property set, having the same length and format: unless they are not comparable and it is no use using GA.

**Population**

The population is a list, containing all individuals. The population might grow or shrink based on the mutation and inheritance. Likely, the populations overall fitness is increased in time, as the better and better individuals (in the population) inherit their parent’s better and better genes.

**Selection and inheritance**

Selection is the method when we sort the individuals, based on their overall fitness calculated by a mathematical model (for example roulette wheel algorithm) and we select the best of them for inheritance. If the list is too short, we will not be able to move enough big in the solution space. If it’s too long, we will waste capacity. When we combine the genes of the selected individuals it is called inheritance. There are several methods for inheritance and several variants of these methods. In this part, new children individuals will born with different gene combinations inherited from their parents.

**Mutation and crossover**

Mutation and crossover are methods for selecting or combining the genes for the children. There are more variations for these methods .These are also mathematical models to model evolutionary biology.

***Summary of GA***

Genetic Algorithm is a very good tool to solve problems in computer science with a high computational complexity. These problems are mainly optimization problems, and there are some basic rules why and when is it worth to apply GA:

- When the solution space is so big, that it is not possible to find a solution with linear programming in relevant time.

- When we face with multi-constraint problems, with lots of constraints.

- When there is a time limit or a resource limit of the problem.

- When we are not looking for the best solution, just looking for an “enough good” one.

- When there is no exact or known algorithm exists for the given problem.

- When we have parallel computers and distributing the problem is hard with linear programming. It can be applied for example as followings.

- Tour selection, tour finding and tour optimization. E.g.: VRP and TSP are problems, where cost minimization is the goal for agents to visit cities with constraints.

1. **Solving the Vehicle Routing Problem with Genetic Algorithm**

***Describing the specified VPR problem***

In the followings, the VRP to be solved is going to be described. This is the problem which has been solved with GA. This problem is a specified VRP, the ‘Vehicle Routing Problem with constraints’.

Dataset

The data is stored in a csv files.

One is JobExecutionTime.csv and the other is TravelTime.csv

JobExecutionTime.csv – File consists of the execution time at each Stop ID

TravelTime.csv -- File consists of the travel time from one Stop ID to other.

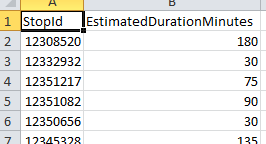
The aim

The goal of this project is to implement a code, which solves the problem as good as possible, under the given circumstances like route length and total time of a route should not exceed the limit.

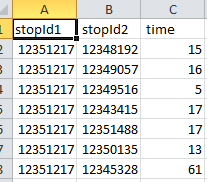
Example Dataset

The given properties are formatted in the following way:

JobExecutionTime.csv



TravelTime.csv



**Pseudo code**

The steps, which were mentioned before can be implemented easily:

Function GeneticAlgorithm()

Initialize\_a\_random\_population\_to\_start\_with();

Evaluate\_individuals()

Repeat

Select\_parents\_for\_mating();

Create\_children\_with\_crossover\_and\_mutation();

Evaluate\_children\_individuals();

Replace\_the\_worse\_of\_the\_population\_with\_children();

Until (Criteria\_is\_not\_reached)\*

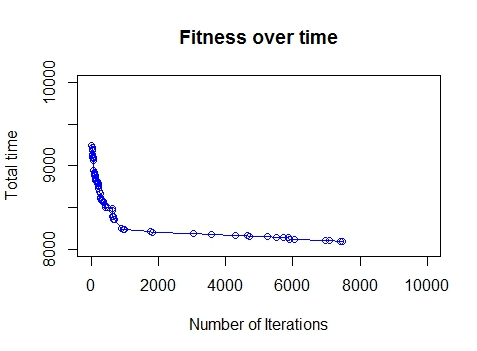
\* The genetic cycle loops, until the given criterion is not reached. This may mean, that we are looking for a solution, enough good for the given problem (a fitness of an individual is lower than a limit), or it might be also a time criteria: the task has ‘T’ time to run, and after T = 0, it is terminated.

**Implementation**

We have used R statistical programming language to implement the algorithm.

**Observations**

I have executed the algorithm for 10000 iterations and Crossover with 0.75, Mutation 0.2 probability accordingly. There is a steady decrease in the total time for the each whole population generated.



**Code**

The below attached is the code implemented to find the optimal routes using Genetic Algorithms.