ORCHID INTERNATIONAL COLLEGE

Tribhuvan University

Institute of Science and Technology

A PROJECT REPORT ON

OPTIMAL ROUTE DETERMINATION USING GENETIC ALGORITHM

Submitted to:

Department of Computer Science and Information Technology

In partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology

Submitted by:

Ningmar Tamang [4982/071]

Rajan KC [4993/071]

Raj Kumar Anamani Shrestha [4994/071]

September, 2018

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to get this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them. We are immensely blessed and would like to extend our profound gratitude to each and every one.

We are highly indebted to Orchid International College for providing us constant supervision, guidance and friendly infrastructure for the successful completion of the project. A special gratitude to our coordinator Mr. Ustab Koirala for his support and motivation.

We are also appreciative of the efforts of our project supervisor Mr. Nipun Thapa and Mr. Dhiraj Jha who guided us throughout the project and provided us for all the necessary ideas and information.

Last but not the least, we are thankful to get encouragement and support from each and every one of our colleagues especially team members, who have made valuable comment suggestions on this proposal which gave us an inspiration to improve our project. We thank all the people for their help directly and indirectly to complete our project.

Ningmar Tamang [4982/071]

Rajan KC [4993/071]

Raj Kumar Anamani Shrestha [4994/071]

Abstract

Optimal Route Determination is a simple interface for viewing the shortest or the near to optimal nodes/points selected by the user. It can be used by various e-commerce sites for providing delivery service seamlessly to their customers, or any particular individual for viewing the optimal route to travel to different destinations. There are various other route determining application available and are developed, some of which uses Google Maps API. With the use of genetic algorithm in this application, traveling through different nodes, which are comparatively complex, are evaluated and near to optimal solution is provided. Genetic Algorithm, a heuristic approach towards solving the route solving problem is presented through this system.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Program Interface
GIS	Geographical Information System
HTML5	Hyper Text Markup Language version 5
SPP	Shortest Path Problem
TSP	Traveling Salesman Problem
GA	Genetic Algorithm

Chapter 1 - Introduction

1.1 Overview

The shortest path problem (SPP) is a problem of finding the shortest path or route from a starting point to a final destination whereas Travelling Salesman Problem is a problem of finding optimal solution traveling from different nodes/cities and finally getting to source from where it starts. Generally, in order to represent the SPP, we use graphs. A graph is a mathematical abstract object, which contains sets of vertices and edges. Edges connect pairs of vertices. Along the edges of a graph it is possible to walk by moving from one vertex to other vertices. Depending on whether or not one can walk along the edges by both sides or by only one side determines if the graph is a directed graph or an undirected graph. In addition, lengths of edges are often called weights, and the weights are normally used for calculating the shortest path from one point to another point. In the real world it is possible to apply the graph theory to different types of scenarios. For example, in order to represent a map, we can use a graph, where vertices represent cities and edges represent routes that connect the cities. If routes are one-way then the graph will be directed; otherwise, it will be undirected. There exist different types of algorithms that solve the SPP such as Dijkstra's Algorithm, Floyd-Warshall Algorithm, Bellman-Ford Algorithm, Genetic Algorithm and different approaches are used like exact approach or non-exact approach.

1.2 Background

The TSP is one of the most notorious in the field of combinatorial optimization, and one of the most well-studied [6]. The study of TSP is believed to be started during 19th century and the study extended from late 20th century as its importance in computing has been recognized. The studies and researches are going on as it is one of NP-complete problem. Held and Karp proposed trying to find a minimum-cost tour in the symmetric case by trying to find an optimal weighted 1-tree [4]. The problems in real time may be different like sometime we may or may not get back to source node at last. So, it is more appropriate to address the problem by assuming we will be returning to

source at last. Nevertheless, work has continued on the TSP through attempts to find polynomial time approximation algorithm. One of the earliest papers on solving the TSP dealt with Euclidean TSP finding a tour through cities of the 48 continental states of U.S. [3]. The Euclidean TSP is NP-complete [2].

1.3 Statement of Problem

Path finding generally refers to finding the shortest or optimal path between any two locations. Many existing algorithms are designed precisely to solve the SPP such as Genetic, Floyd-Warshall algorithm. Among all the methods present, we will be using genetic algorithm to find the solution. There are many algorithms to find the shortest path. Many problems may appear in real world environment like road on construction or traffic jam and so on. It is important to study the suitable algorithm for different scenarios and implement according to it to provide the best possible method to find the optimal effective path so as to reduce time and cost in any way possible. There are many applications of route determination, once the effective path is found, such as finding cost effective routes for logistic purposes, network designs etc.

1.4 Objectives

The main objective of this project is to implement genetic algorithm to find the optimal path. The path generated by this application can be ultimately used for road routing in real world environment which can be used for effective and efficient delivery of goods. The main goals and objectives of this project can be summarized into the following points:

- To find the optimal route that passes through all the nodes/points.
- To minimize the transportation cost and also not to visit the same place twice except for the source.
- To uplift the delivery logistics of ecommerce companies by providing an efficient route to deliver products to customers.

1.5 Scope of the Project

This application is a web-based application, which aims to find the optimal route from the selected points. This application is suitable for business purposes such as e-

commerce companies, delivery logistics, and even for personal use also. This application requires a connection to the internet to perform its operations.

1.6 Literature Review

The TSP is a classic tour problem in which a hypothetical salesman must find the most proficient sequence of destinations in this territory, stopping only once at each, and ending up at the initial starting location. Application of the TSP is not limited to finding the path for different cities but has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. The most popular technique in evolutionary computation research has been the genetic algorithm. [7]

In this project, genetic algorithm has been used to evaluate the collection of genes i.e. chromosome and to select the best of them. The chromosomes represent the collection of nodes or cities. General GA performs basic operations of selection, crossover and mutation and so it is implemented in this project with some simple modification.

The individuals are initialized with random chromosomes. The number of individuals is total individuals. Also, population consists of array of individuals and each individual contains the collection of cities. In crossover, the population goes through tournament selection which selects only certain number of individuals. The selection is performed randomly. Fittest score of all the selected individuals are evaluated and then the individual with higher fittest value is selected for individual crossover.

In this algorithm, only one parent (individual) is selected for crossover rather than two parents (individuals) because every individual has all the information to generate best solution. Thus one best individual is selected rather than two parents.

Fittest score is calculated by fittest function which is based on total distances between the cities of individual. A fittest function is inversely proportional to the total distances i.e. $f = 1/d$; in individual crossover half of individual's genes transforms to new children and other half is randomly assigned and the fittest score is calculated.

The step after evaluating the fittest score is mutation which takes place in the new children. In order for a mutation to occur, mutation rate is compared with random number generated by browser. The mutation rate must be compared with random values between 0 to 1 (0 <= Math.Random() <1). If the condition remains true, mutation is performed in the new children. It occurs in every single generation with same process to give us the best solution possible among all the generation.

Chapter 2 - Requirement Analysis and Feasibility Study

2.1 Overview

Optimal Route Determination is one of the most evaluated system for obtaining the cost effective route through several points. Before starting the development of the new system, different requirements are taken into consideration and an analysis of feasibility of our system is performed.

2.2 Requirement Specification

Requirements of our system can be broadly classified into two categories, i.e. functional specification (which specifies the behavior or function that a system should perform) and non-functional specification (which describes how a system should behave) which are stated below.

2.2.1 Functional Requirements

a. **System Requirements:** Being a web-based application, our system can be operated on almost any operating system having the latest web browser installed into their system.
b. **Input/Output:** The user of our system must provide points to be travelled in a particular boundary provided by the system.
c. **Control Requirements:** Alerts are given to the system when the user doesn't select any points within the boundary.

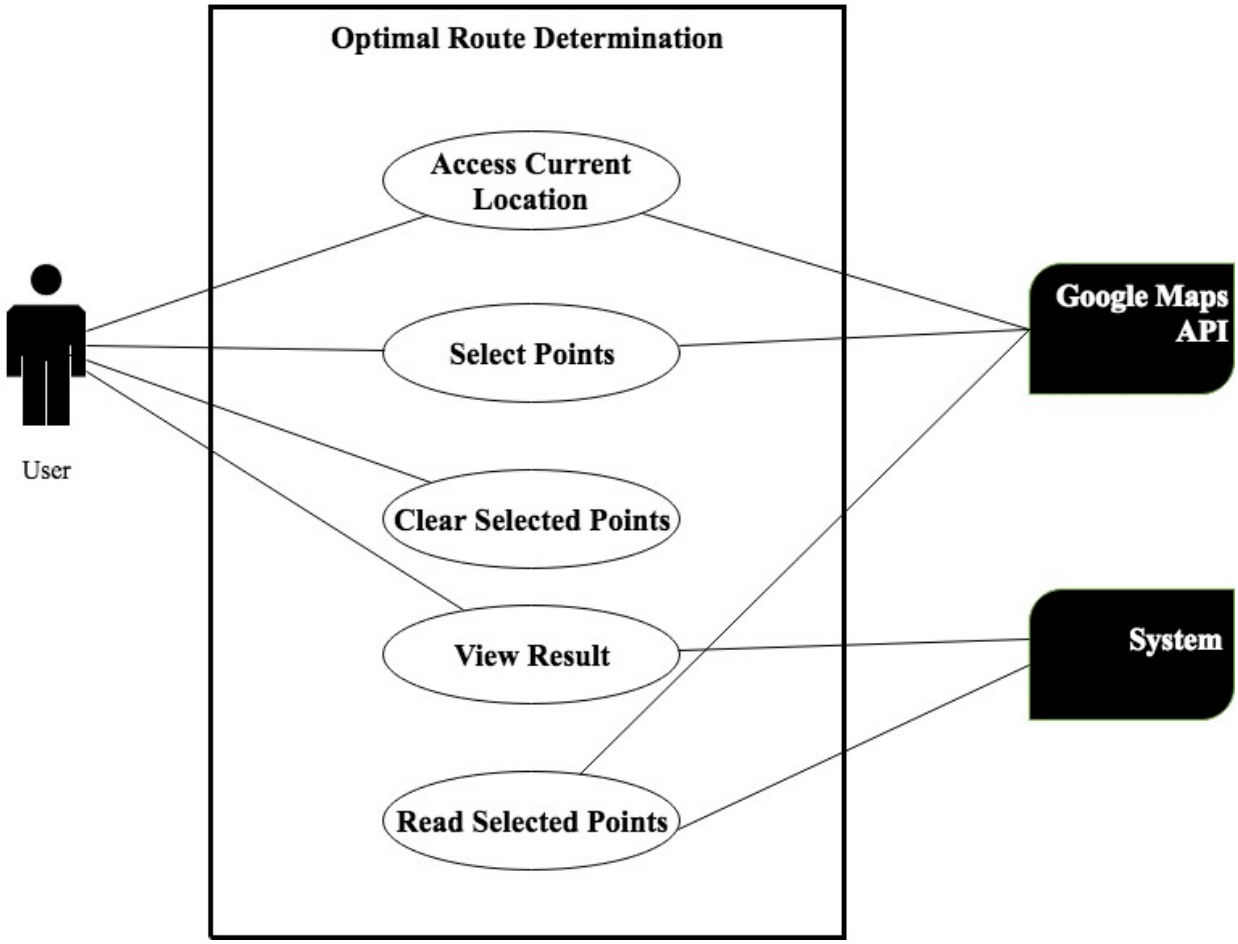

Figure 2.1: Use Case Diagram

Use Case Description

Table 2.1: Access Current Location

Use Case ID	UC-1
Use Case Name	Access Current Location
Primary Actor	User
Secondary Actor	Google Maps API
Description	User access the current location in the map
Pre-Condition	Connected to internet and browser opened
Post-Condition	Valid current location of user

Table 2.2: Select Points

Use Case ID	UC-2
Use Case Name	Select Points
Primary Actor	User
Secondary Actor	Google Maps API
Description	User selects specific points in the maps
Pre-Condition	Google Maps API working properly
Post-Condition	Update the values of the selected points

Table 2.3: Clear Selected Points

Use Case ID	UC-3
Use Case Name	Clear Selected Points
Primary Actor	User
Description	User clears all the selected points from map
Pre-Condition	User needs to selects various points in map
Post-Condition	Check whether only one or no point selected

Table 2.4: View Result

Use Case ID	UC-4
Use Case Name	View Result
Primary Actor	User
Description	User views the result
Pre-Condition	More than one points selected

Table 2.5: Read Selected Points

Use Case ID	UC-5
Use Case Name	Read Selected Points
Primary Actor	Google Maps API
Description	Maps API reads the selected points on the map
Pre-Condition	User need to select more than one point
Post-Condition	None

2.2.2 Non-Functional Requirements

a. **Extensibility:** Various features can later be added into this application in order to increase the efficiency of the system.
b. **User Friendliness:** Interaction and navigation of this system is user friendly, which makes naïve user to access the features of the system.
c. **Performance:** The system provides the appropriate result regarding the route with effective performance.

2.3 Feasibility Analysis

Feasibility study is the study that incorporates and idea of whether a system is feasible or not. A system is feasible if it is cost effective to develop, install, operate and maintain. Our system mainly focuses on following three feasibility analysis categories.

2.3.1 Technical Feasibility

Our system is a web application, which is supported by almost all the latest web browsers and modern technologies such as HTML5 and JavaScript, which makes our application technically feasible.

2.3.2 Operational Feasibility

Due to the fact that our system becoming a web application, it can be easily operated in any devices having web browsers (supporting HTML5) installed into their system. In addition, the user of our system can easily use the system without internet connection.

2.3.3 Economic Feasibility

The system is a web based application and does not require large amount of capital to develop. The development process can be easily performed is any computer or laptop.

Chapter 3 - System Analysis and Design

3.1 Overview

System analysis and design is a process through which requirements are translated into a representation of software. Initially the representation depicts a holistic view of software. Subsequent refinements lead to a design representation that is very close to the source code. System analysis and design serves as the foundation for all software engineering and software maintenance steps that follow.

3.2 Methodology

The purposed system is a web-based system which focuses on determining optimal route using genetic algorithm. This system contains a webpage that allows user to view the best route dynamically. In order to use this algorithm there must be a set of vertices or nodes in which the cost is evaluated and provide near to optimal solution.

3.3 Algorithm Used

Genetic Algorithm is heuristic search algorithms based on evolution of natural selection and genes of the parents. Although this algorithm may not provide the fastest result, it provides the near to optimal solution.

The basic process of a genetic algorithm is as follows:

i. Initialization: Creating an initial population. This population is generally generated at random and can be of any size, from a few individuals to thousands.

ii. Evaluation: Each member of the population evaluates and calculates a "physical condition" for a particular individual. The value of the physical condition is calculated according to its adaptation to the requirements. These requirements can be simple or complex.

iii. Selection: Constant improvement of physical condition is performed in this stage. Selections helps to reject bad individuals and obtain only the best individual in the population.

iv. Crossover: New individual that combine aspects of our selected individuals are created during crossover. With the combination of certain characteristics of two or more individuals, an even more fertile offspring that will inherit the best qualities of each of its parents are obtained.

v. Mutation: Process of including certain randomness to the genetics of the populations to obtain different results is mutation. The randomness may be tiny according to the need.

The process is repeated until a completion state is not accomplished.

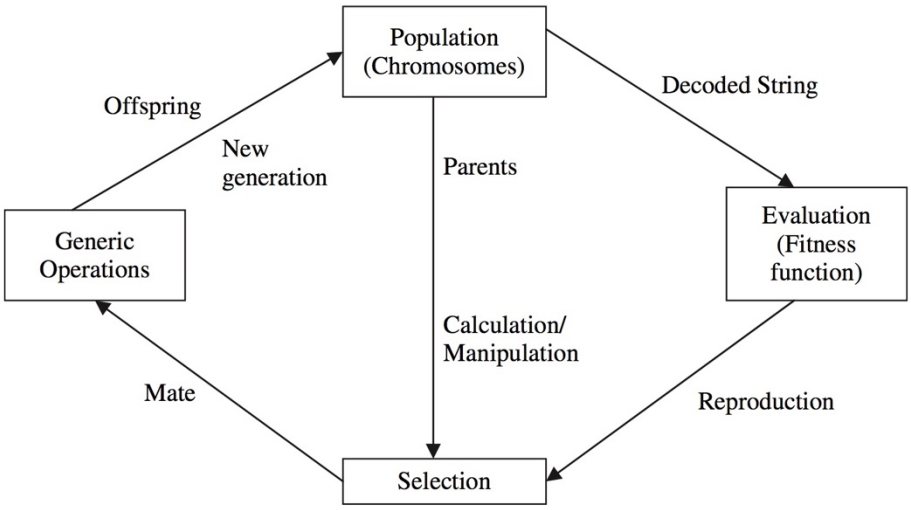

Figure 3.1: Genetic Algorithm Cycle

In this application, genetic algorithm is adopted to determine the optimal route. The idea or base for the project is same like initialization, selection with genetic operators till the solution is found. However, some technique has been used; mostly of randomness during selection and mutation. The pseudo codes mixed explanation of the algorithm implementation is as follows:

1. Initialization: the random individuals are initialized. The population is created from individuals who contain random initialization of chromosomes. These chromosomes store information of cities.
2. For generation = 1 to maximum number of generation selected

3. Evolve population //answer or set of solution is produced here. Evolving does crossover and mutation both.
4. Crossover in the population to generate new offspring

i. From the population each individual is checked for fittest function (f = 1/d)

ii. The high fittest individual is selected from population with elitism true, which does not happen unless we do it. If elitism is true and the random population with highest fitness value is added in new population.

iii. This occurs if elitism is false or fittest is not found then select population with tournament selection where population has random individuals and gets the fittest value among them and do crossover to that individual to create new offspring and new population.

The individual chromosome is randomized such that no cities are recorded twice and add to new offspring chromosomes and if any cities or genes is excluded from parent then add it. Adding chromosome to new offspring it does it randomly but pass down certain chromosome information. The selected offspring is selected for new population

5. Mutation: For each individual in new population, Mutation is randomized and the most common mutation is inversion and shift operator but we randomized the chromosome similar as in crossover keeping some chromosomes information same. If this gives us fittest set of chromosome, then new set is taken. Mutation rate affect mutation if it is high there is going to be more mutation and low mutation with low mutation rate.
6. Best offspring is selected and goes to next generation.

3.4 System Diagrams

Design is the first step in the development phase for an engineered product or system. Without design, there is presence of risk in building an unstable design-one that will fail when small changes are made, one that may be difficult to test, and one whose quality cannot be accessed until late in the software engineering process.

Taking software requirements specification document of analysis phase as input to the design phase we have drawn Unified Modeling Language (UML) diagrams. UML depends on the visual modeling of the system.

3.4.1 Activity Diagram

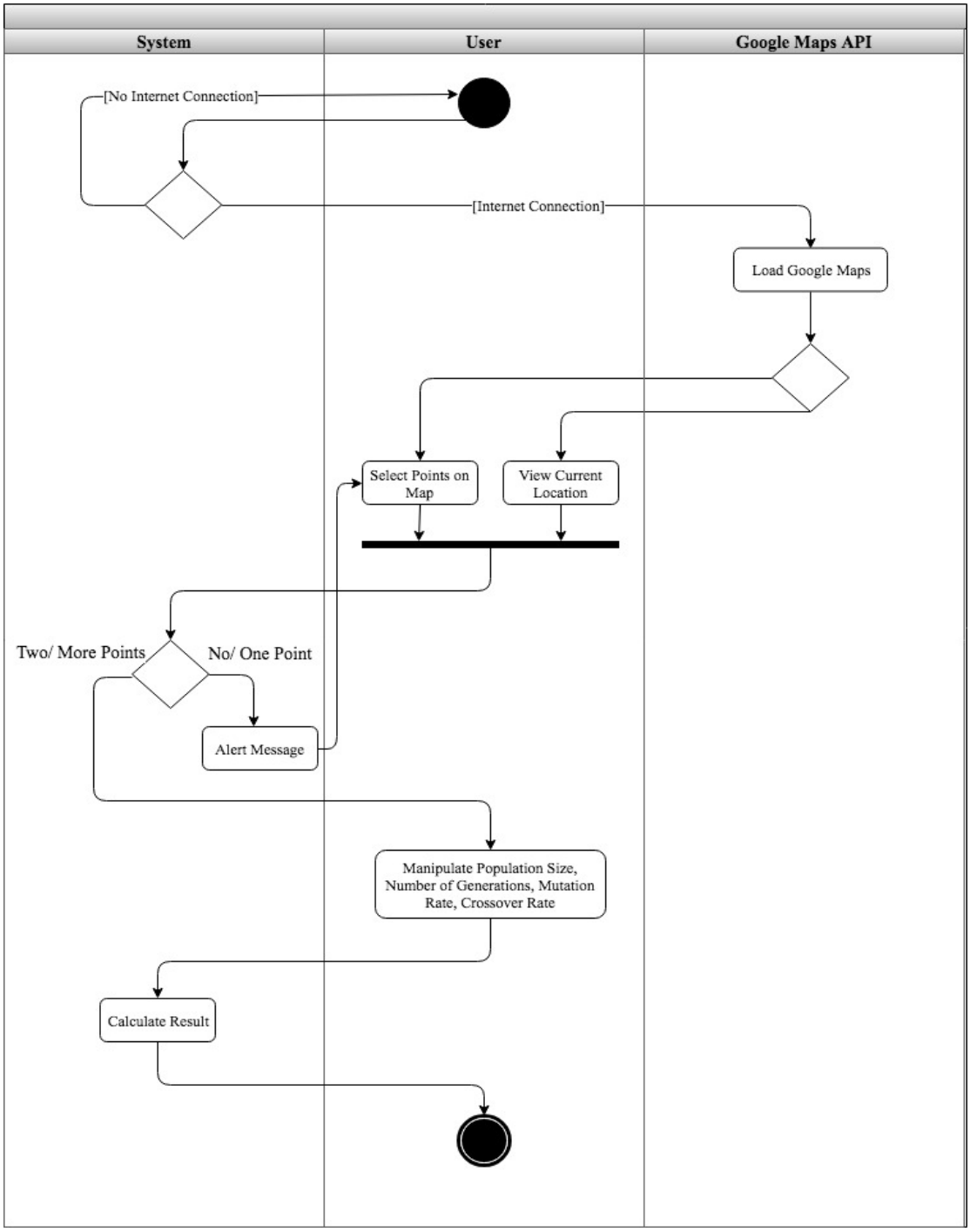

Figure 3.2: Activity Diagram

3.4.2 State Diagram

State diagram displays the sequence of states that an object of an interaction goes through during its life in response to received stimuli, together with its responses and actions.

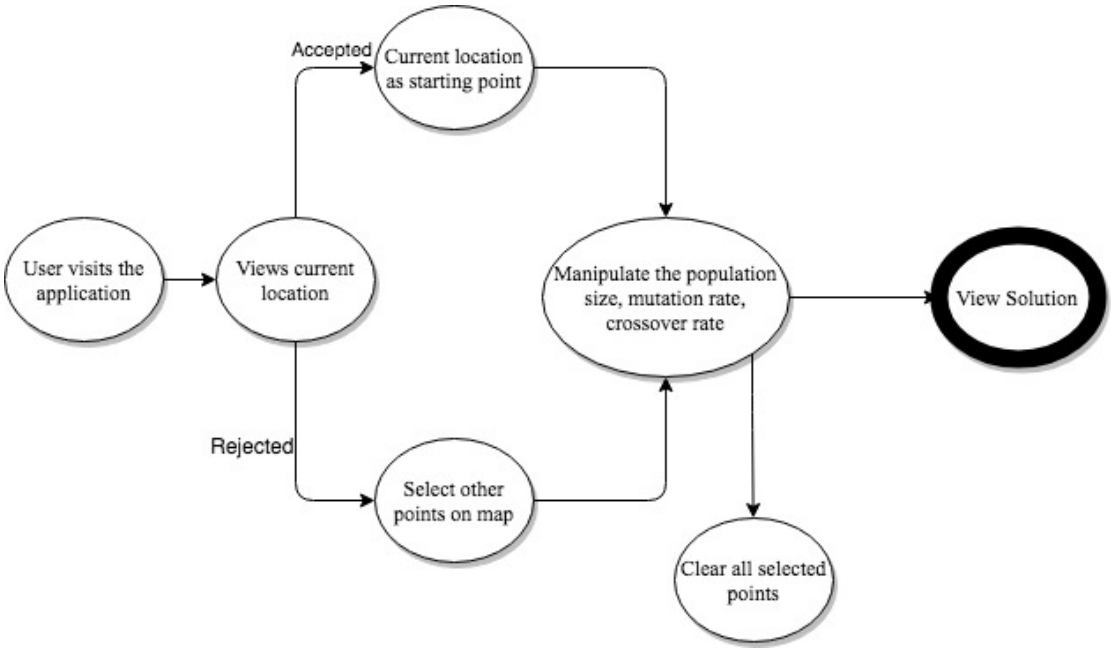

Figure 3.3: State Diagram

3.4.3 Sequence Diagram

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of vertical dimension (time) and horizontal dimension (different objects).

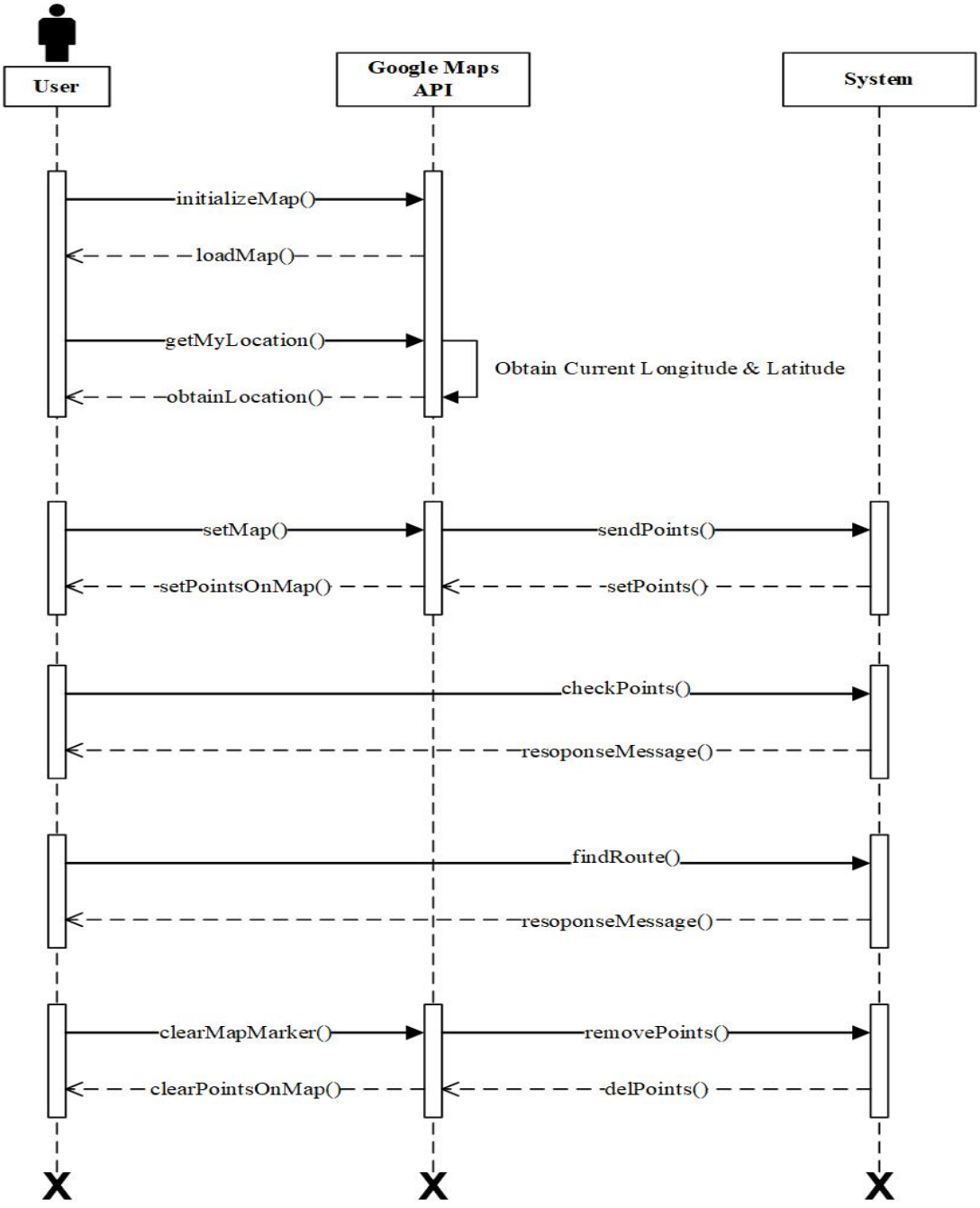

Figure 3.4: Sequence Diagram

Chapter 4 –System Implementation and Testing

4.1 Implementation

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively. After the successful development of the system, the system is implemented in real world environments where there may be present various exceptional situations and conditions which is then tested. The purpose of this system is to implement the genetic algorithm for the given set of points provided by the user and provide an effective outcome.

Variation wins in GA, so parameters which gives more variations provide effective results. Some parameters used in this projects are discussed below:

Population size: Population is collection of individuals. The population size gives the number of individual produced in each generation. The more the population size, more chance of creating individual with new set of chromosomes. It affects the time taken to travel the nodes as well. As the population size grows, the time to make new individuals grow in indistinct number.

Mutation rate: Mutation rate is compared to random number generated by browser usually 0-1 (inclusive 0 but not 1). In each generation, if mutation rate is less than random then mutation is performed, giving new possible set of chromosome from the selected children. Certain dilemma occurs due to random comparison. Despite some dilemma, it gives satisfying result.

Generation: Generation is the number of time the algorithm has to run causing a loop. More the generation, more the number of new population and individuals are created. It does tradeoff with time as more generation may create new individuals, but it takes more time.

Classes, variables and methods used while implementing the genetic algorithm in this project are discussed below:

```
ga = {
        "mutationRate"
        "populationSize"
        "tournamentSize"
        "generations”
        "getConfig": function() {}
        "evolvePopulation": function() {}
        }
population = {
                individuals = [];
                this.initialize = function() {};
                this.mutate = function() {};
                this.crossover = function() {};
                this.addIndividual = function(individual) {};
                this.getFittest = function() {};
                this.getBest = function() {};
                }
individual = {
                chromosomeLength;
                fitness;
                chromosome;
                this.initialize = function() {};
                this.setChromosome = function() {}
                this.mutate = function() {}
                this.getDistance = function() {}
                this.calcFitness = function() {}
                this.crossover = function() {}
                }
```

Some of the major services of Google Maps API used in this project are also discussed below:

google.maps.Map()

google.maps.event.addListener()

google.maps.Marker()

google.maps.DistanceMatrixService()

google.maps.DirectionService()

google.maps.DirectionRender()

4.2 Coding Tools

Programming tools that are used to create, debug, maintain or support the system, mainly used by developers are known as coding tools. Tools can be regarded in majorly two types i.e. front end and back end tools. These tools are combined together to form a logically stable application. Front end generally involves the user interface and the user interaction with the system and back end involves the development of database and the actual program or application logic. By combining these two factors, we can create a full stack web application. Our system is also a web application and different tools are used during the development of this project.

4.2.1 Front End

The interface which allows the user of the system to interact with the actual system is known as front end. Generally, the front end of a system must be easy to navigate and easy to interact with the application. In our project, we have used the following set of front end tools:

- JavaScript
- HTML5
- CSS3
- Chart.js

4.2.2 Other Tools and Platforms

1. Atom

Atom is an open-source text editor for macOS, Linux and Windows OS, which has Git Control embedded in it. Built using web-technologies, Atom is a desktop application written in CoffeeScript and Less. This entire application was written in HTML5, CSS and JavaScript using Atom. It was also used for compiling, testing, debugging and executing the source code.

2. GitHub

GitHub is a web-based hosting service for version controlling and code sharing. It provides features such as bug tracking, task management and collaboration between the group members. As our member were located at different residential locations, so we chose to use GitHub for maintaining, sharing and tracking code. Every member would commit certain changes, push the bug-free code into the repository and pull to make various changes.

4.3 Testing

The testing process focuses on the logical intervals of the software ensuring that all statements have been tested and on functional interval is conducting test to uncover errors. In other words, software testing is the process of evaluation of a software system item to detect differences between given input and expected output. Software testing is a process that should be done during the development phase to assess the quality of the product.

Table 4.1: Test Cases throughout the project

S.N.	Test Case ID	Test Description	Outcome	Pass/ Fail
1.	TC-1	No points are provided	Alerts the user with appropriate message	Pass

2.	TC-2	Only one point is provided	Alerts the user with appropriate message	Pass
3.	TC-3	Only two points are provided	Provides the result for the given two points	Pass
4.	TC-4	More than 2 points and less than 9 points are provided	Provides the result for the given points	Pass
5.	TC-5	More than 9 points are provided	Alerts the user with appropriate message	Pass
6.	TC-6	Loading Google Maps API	Google Maps API loaded in the application	Pass
7.	TC-7	Getting the current location from Google Map	Current location coordinate of the user is obtained	Pass
8.	TC-8	Setting minimum value of population size, mutation rate and number of generation	Less variation providing ineffective result	Pass
9.	TC-9	Setting maximum value of population size, mutation rate and number of generation	More variation providing efficient result	Pass

Chapter 5 - Conclusion

This application provides a convenient way to obtain the optimal route by selecting different points/nodes in a boundary that helps user to travel according to it. The application is inspired from Geographical Information Systems (GIS) such as Google Maps, which captivated us to build it from scratch in JavaScript.

5.1 Limitations

The optimal route determination is itself a complex problem which does not have a particular solution and it also needs big computational capacity along with new location, the time to calculate the optimal solution increase exponentially.

- Real time traffic is not available in our system.
- Routes obtained may not be shortest always.
- Due of free version of Google Maps API, the number of nodes/points must be less than 10 according to Distance Matrix Service.

5.2 Future Enhancements

5.2.1 Adding Real Time Environment Variables

Real Time Environment variables such as traffic in a particular route, weather condition, road condition, vehicle efficiency can be mapped and added in the system for better outcome.

5.2.2 Implementation of other algorithms

Algorithms such as Dijkstra Algorithm combined with Genetic Algorithm may provide a hybrid algorithm which may generate more accurate results in less time.Other algorithms like Ant-Colony Optimization Algorithm, Neural Network can also be implementedto obtain a robust and near to accurate results.

References

[1] J. Bang-Jensen, G. Gutin and A. Yeo, *When the greedy algorithm fails*. Odense: University of Southern Denmark. Department of Mathematics and Computer Science, 2004, pp. 121-127.

[2] C.H. Papadimitriou, *The Euclidean traveling salesman problem in NP-complete*, 1977, pp. 237-244.

[3] G. Dantzig, R. Fulkerson and S. Johnson, *Solution of a large scale traveling salesman problem*. Ft. Belvoir: Defense Technical Information Center, 1954, pp. 119-125.

[4] M. Held and R. Karp, *The traveling-salesman problem and minimum spanning trees*, 1970.

[5]M. Berndtsson, J. Hansson, B. Olsson and B. Lundell, *Thesis projects*, 2nd ed. London: Springer, 2008.

[6] E. L. Lawler, J. Lenstra and C. Hurkens, *The Traveling salesman problem*. Wiley, 1985.

[7] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*. Berlin: Springer, 2008.

[8] R. Haupt and S. Haupt, *Practical genetic algorithms*, 2nd ed. Hoboken, N.J: Wiley-Interscience, 2004.

[9] M. Konarski and W. Zabierowski, *Using Google Maps API along with technology .NET*, *2010 International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, Lviv-Slavske, 2010, pp. 180-182.

[10] C. Dawson, *Projects in computing and information systems*, 2nd ed. New York: Pearson Prentice Hall, 2009, pp. 181-214.

Appendices

A. Screenshots

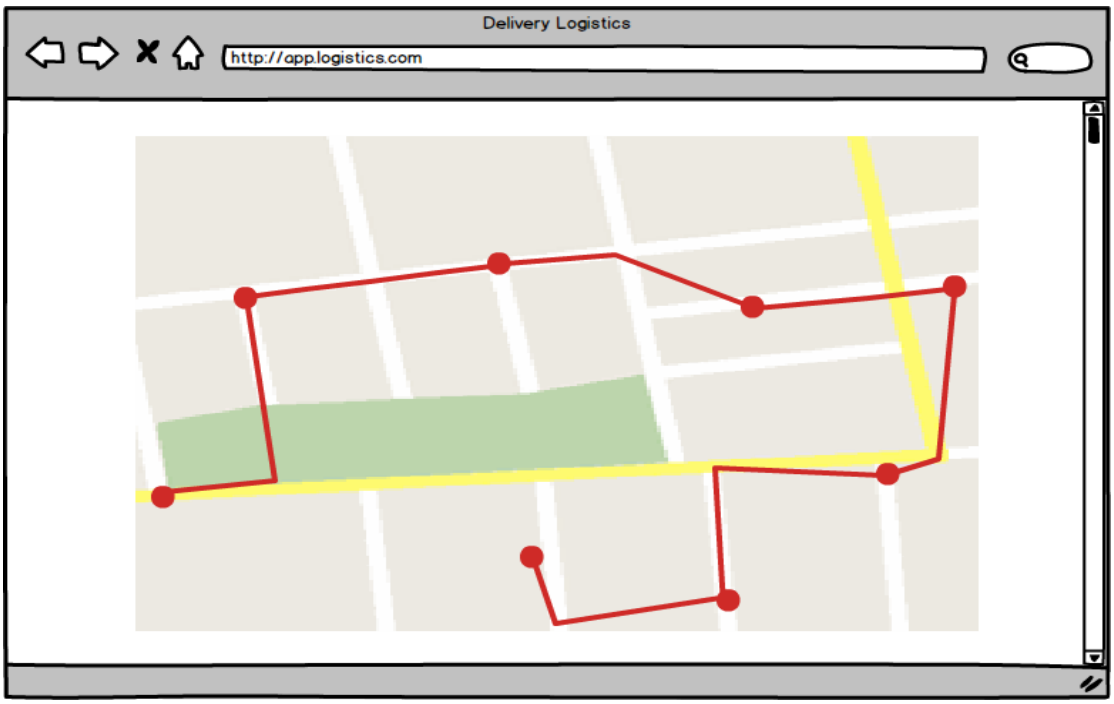

Figure 5.1: Wireframe of Route Determination

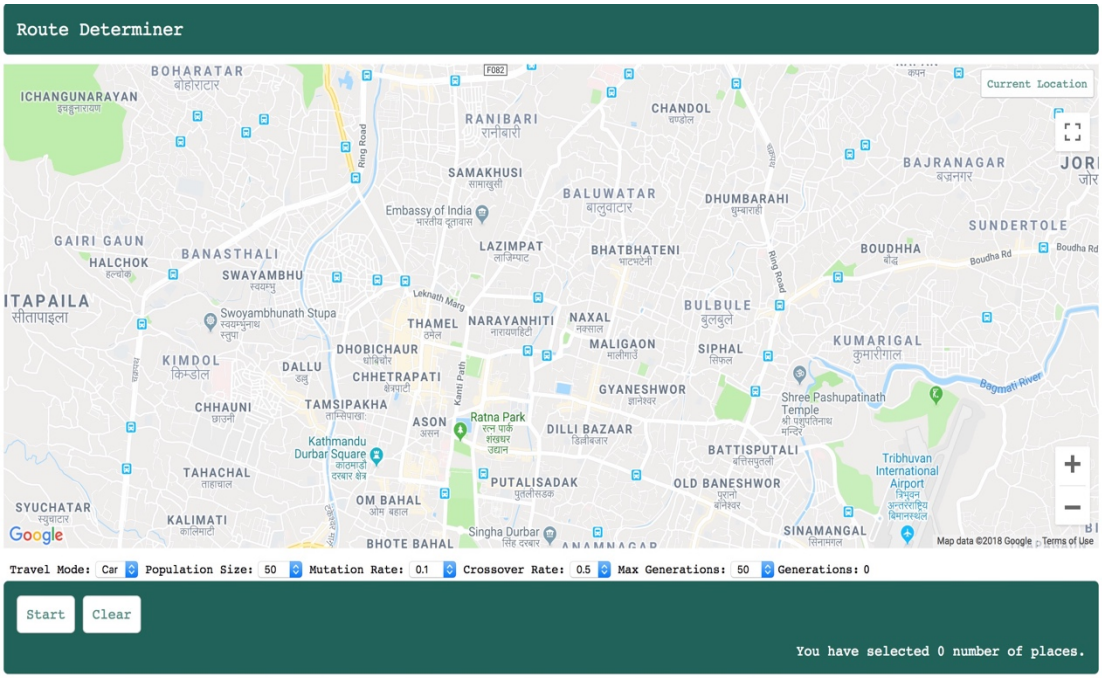

Figure 5.2: Screenshot 1 (Landing Page)

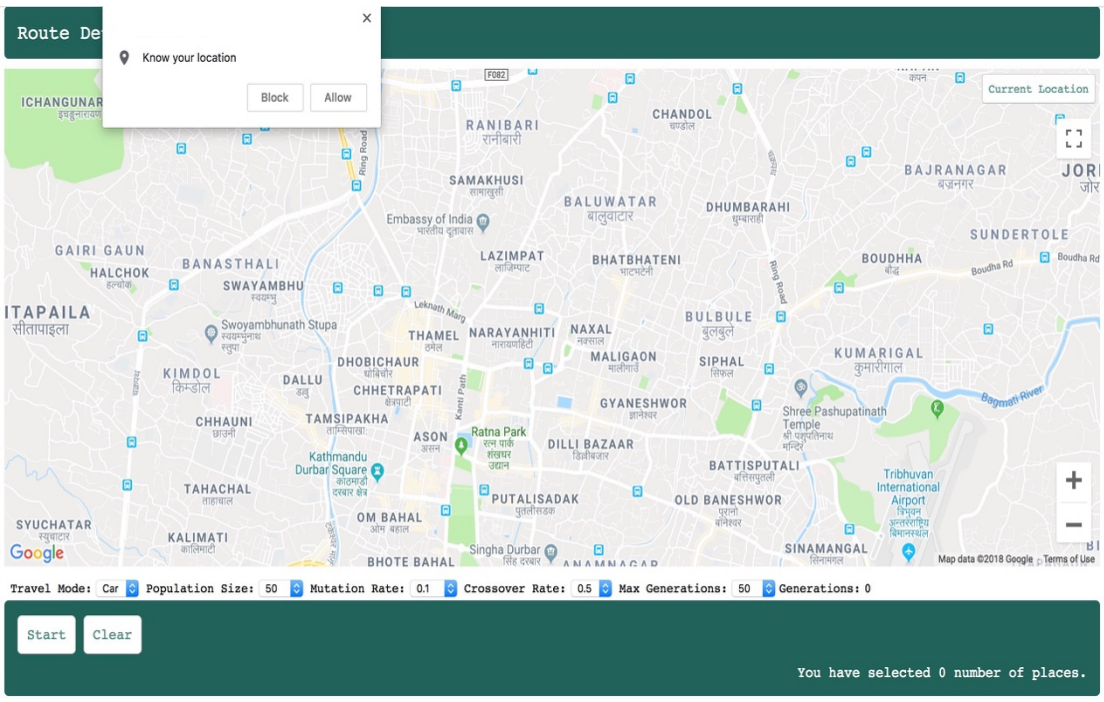

Figure 5.3: Screenshot 2 (Alert for current location access)

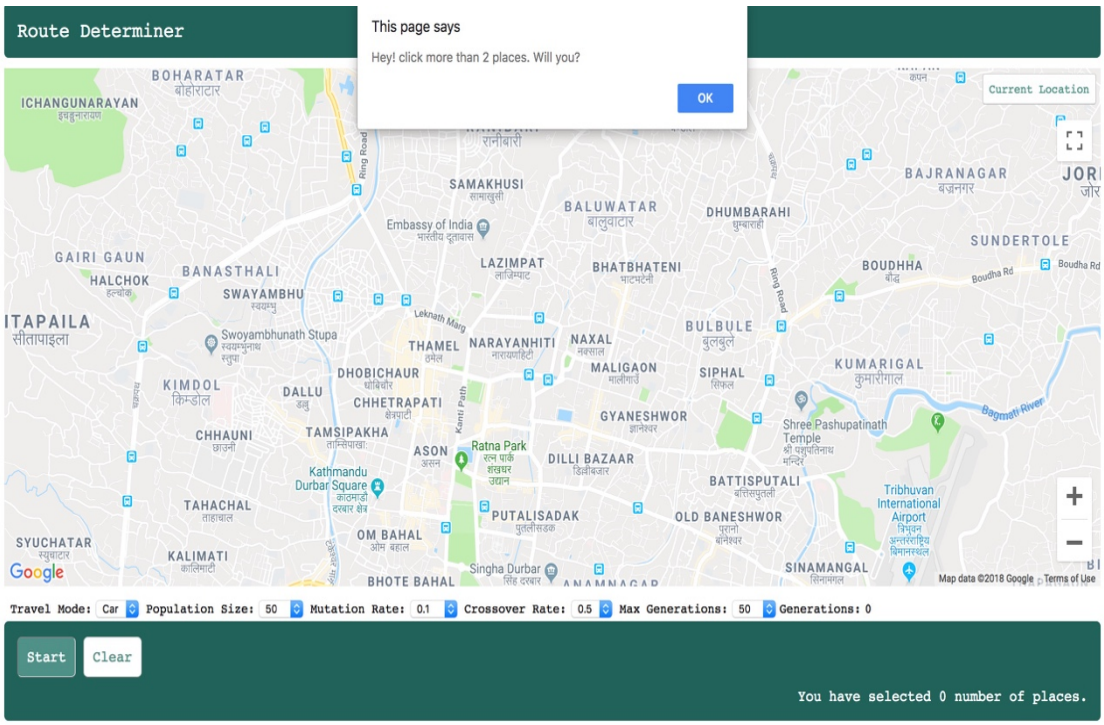

Figure 5.4: Screenshot 3 (Alert for no/one point selected)

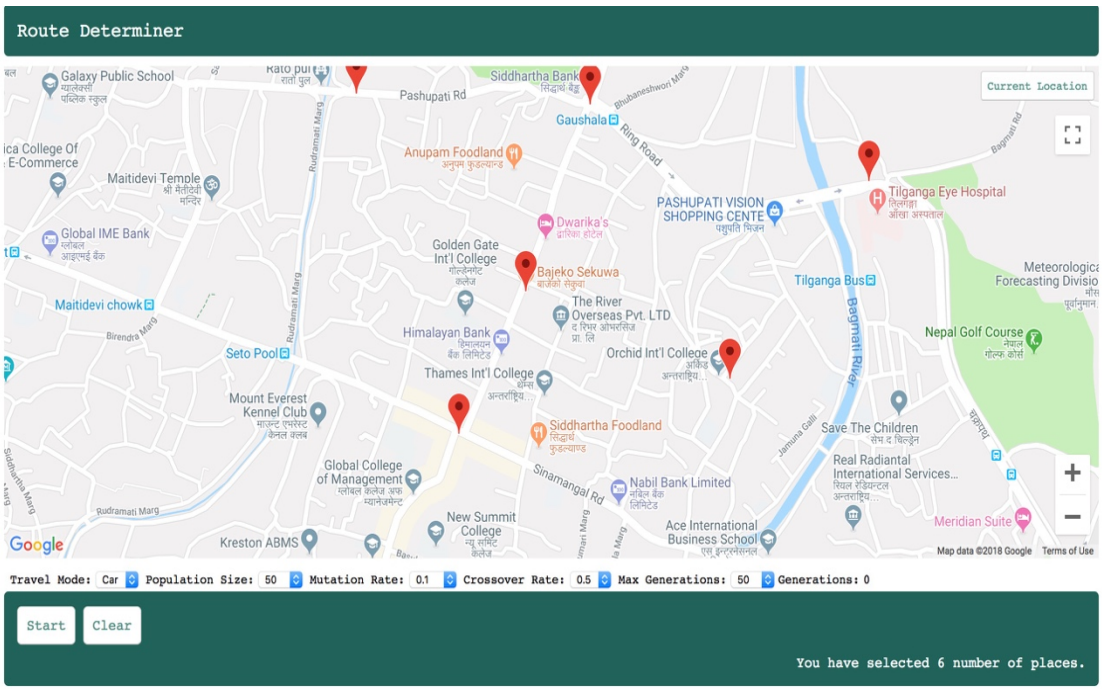

Figure 5.5: Screenshot 4 (Adding marker)

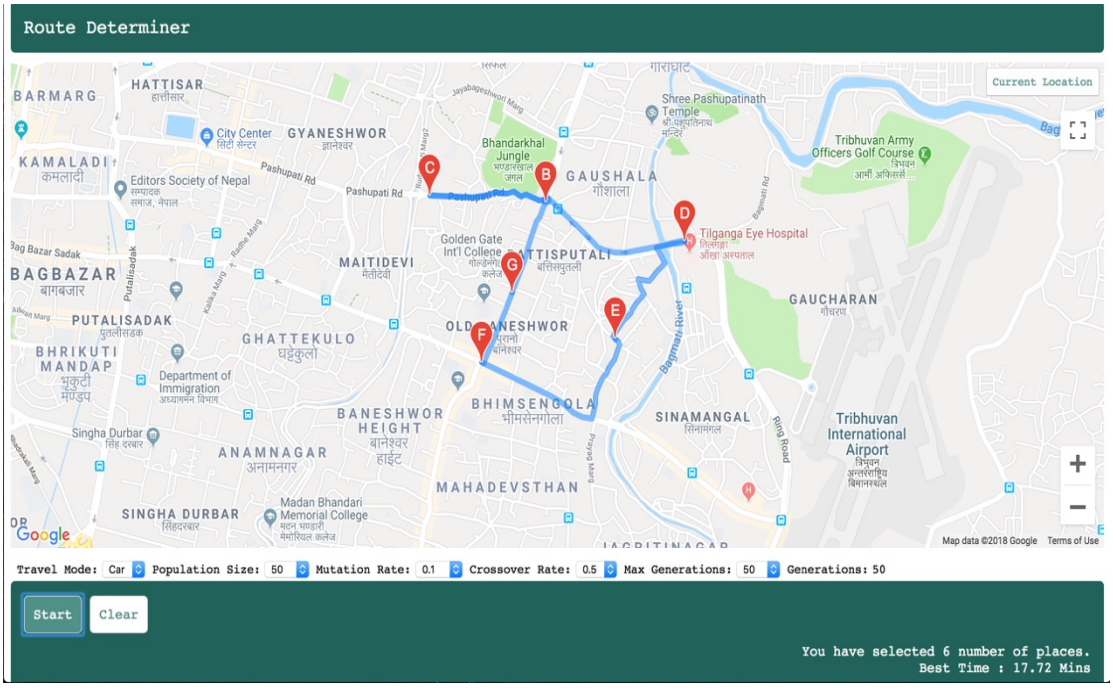

Figure 5.6: Screenshot 5 (Optimal Path Generation)