Optimal Traffic Light Signaling Based on Genetic Algorithm Approach

   Alok    Yadav                 58090001

   Tan             Tianyu    58090005

Sajeerat Aussavaruengsuwat 58090036

Submitted to the Software Engineering Program of the International College  
in partial fulfillment of the requirements for   
Team Software Project

at the

KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG

October 2018

COPYRIGHT 2018

INTERNATIONAL COLLEGE

KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG

Table of Contents

[Abstract 3](#_Toc529877762)

[Introduction 3](#_Toc529877763)

[Chapter 1: Objectives and Problem Description 4](#_Toc529877764)

[*1.1 Problem statement 4*](#_Toc529877765)

[*1.2 Problem Description 4*](#_Toc529877766)

[*1.3 Objectives 4*](#_Toc529877767)

[Chapter 2: Literature Survey 6](#_Toc529877768)

[Chapter 3: Background Knowledge 7](#_Toc529877769)

[*3.1 Traffic Simulation 7*](#_Toc529877770)

[3.1.1 Types of Simulation Models 7](#_Toc529877771)

[*3.2 Traffic Simulation Framework 8*](#_Toc529877772)

[3.2.1 Nagel-Schreckenberg model 8](#_Toc529877773)

[3.2.2 TSF model 9](#_Toc529877774)

[3.2.3 Traffic Simulation Framework 9](#_Toc529877775)

[*3.3 Genetic Algorithms 11*](#_Toc529877776)

[3.3.1 Concept 11](#_Toc529877777)

[3.3.2 Terminology 11](#_Toc529877778)

[3.3.4 Computing process 11](#_Toc529877779)

[3.3.5 Problem domain 12](#_Toc529877780)

[3.3.6 Disadvantage 12](#_Toc529877781)

[*3.4 Current State of Traffic Control in Bangkok 12*](#_Toc529877782)

[Chapter 4: Approach 13](#_Toc529877783)

[*4.1 Setting up traffic simulation model: 13*](#_Toc529877784)

[*4.2 Long term genetic algorithm (GA1): 14*](#_Toc529877785)

[*4.3 Short term genetic algorithm (GA2): 16*](#_Toc529877786)

[*4.4 Experiments 17*](#_Toc529877787)

[Chapter 5: Development/Implementation 19](#_Toc529877788)

[*5.1 Data gathering 19*](#_Toc529877789)

[*5.2 Using traffic data in simulation 20*](#_Toc529877790)

[*5.3 Genetic Algorithms and Traffic Simulation 20*](#_Toc529877791)

[*5.4 GUI application 22*](#_Toc529877792)

[Chapter 6: Experimental Result 23](#_Toc529877793)

[Chapter 7: Current Progress & Project Plan for Next Term 24](#_Toc529877794)

[References: 25](#_Toc529877795)

List of figures

**Chapter 3:**

[Figure 3.1: Continuous Model, taken from [13]………………………………………………………..7   
Figure 3.2: Discrete Model, taken from [13] 7](#_Toc529738617)

[Figure 3.3: Editing simulation parameters and starting simulation, taken from [8] 10](#_Toc529738618)

[Figure 3.4: The main window of the TSF software, taken from [8] 10](#_Toc529738619)

**Chapter 4:**

[Figure 4.1: Diagram showing the operation of proposed method 14](#_Toc529804824)

[Figure 4.2: Genotype with 3 crossroads and 2 time steps in GA1 15](#_Toc529804825)

[Figure 4.3: Genotype with 3 crossroads in GA2 17](#_Toc529804826)

**Chapter 5:**

[Figure 5.1: class diagram of genetic algorithms 22](#_Toc3980044)

[Figure 5.2: Sequence diagram of using genetic algorithms 23](#_Toc3980045)

[Figure 5.3: Sequence diagram of getting fitness values 23](#_Toc3980046)

# Abstract

This report presents a methodology for overcoming the computational complexity issue of using genetic algorithms to optimize traffic signal timings over long periods of time. The methodology relies on formulating a long-term traffic control strategy by using a genetic algorithm and traffic simulation model to compute fitness and then uses another genetic algorithm for real time control based on the precomputed strategy.

Keywords: Genetic algorithm, traffic simulation, traffic congestion, computational complexity

**Introduction**

Traffic congestion is a globally occurring problem that results in a massive waste of resources. More than often, the congestion is repetitive and occurs as a result of the increase in inflow from certain sources. Current traffic management methods are far from efficient. As of right now, in Bangkok, the traffic signal timings are constant. They don’t change depending on the traffic conditions. Managing traffic signal timings based on traffic conditions is an efficient way of reducing traffic congestion. Genetic algorithms can be used to optimize traffic signaling based on changing traffic conditions and have been shown to be more efficient compared to other evolutionary strategies [6].

There have been many researches that attempt to use genetic algorithms to optimize traffic signal timings. However, they mostly focused on real time control based on existing traffic conditions. Due to the time constraints of real time control, the simulation time for most of the researches was only of the order of a few minutes. Actions on traffic signals can have long term consequences that are overlooked when using short time simulations. Furthermore, this approach fails to capitalize on the history of traffic, which can help in managing traffic more efficiently. The difficulty of optimizing traffic signaling over a long period of time arises due to the computational difficulty of running repeated simulations over long durations of time over a large area. It has been shown that expanding the area considered yields better optimization results [4], and increasing simulation time will allow the genetic algorithm to find better solutions. However, the potential of using genetic algorithms for traffic optimization hasn’t fully been realized due to the time complexity issues.

In our project, we attempt to overcome this obstacle by creating generic traffic light signal timings for an area during certain periods based on historic traffic data. Furthermore, we are going to be optimizing sequences of signal timings, this way, the timings respond to the way that traffic changes. We use these settings as a basis for future optimization, i.e. use these settings to evolve settings to fit future scenarios. This gives us the benefit of conducting longer simulations to optimize traffic, without having to conduct such long simulations during real-time control. During real-time control, we generate new timings based on the generic timings using much shorter simulations, hence improving computation time.

# Chapter 1: Objectives and Problem Description

## **1.1 Problem statement**

Genetic algorithms can be used to efficiently optimize traffic signals and reduce traffic congestion; however, the time complexity of the task is too high to be practical.

## **1.2 Problem Description**

One of the most cost-effective ways of managing traffic is optimizing traffic signal timings. Genetic algorithms can be used to find near optimal traffic signal timings. However, prior researches on the application of genetic algorithms to the traffic setting problem have had limited results due to the computational complexity of simulating traffic. As a result, optimizing traffic over long periods of time has proven to be problematic. This project aims to overcome this obstacle by performing the computations for long term traffic control prior to real time control and formulating a traffic control strategy. Then using a real time algorithm to adapt to the pre computed long term strategy. The application of this method is expected to reduce the total waiting time of cars by more than 20% for the area considered.

## **1.3 Objectives**

We have established the following major objectives for our project:

**Gather travel speed data for roads in the selected region:**

This data is going to be used to initialize and alter the simulations, as well as to understand the variability of traffic, which we will attempt to simulate.

We are going to acquire the travel speed data at roads near intersections using the TomTom traffic API. Then we will have to use that data to derive information about traffic flow at different time intervals. This information is meant to guide the traffic flow during simulations.

**Set up traffic simulation model:**

In order to use evolutionary algorithms, we need a mechanism to predict the fitness of a solution. We are going to need a traffic simulation model that can accurately predict traffic conditions based on changing traffic inflow and changing traffic light signaling.

Thus far, we have acquired permission to use the traffic simulation framework developed by Pawel Gora. Traffic simulation framework is an advanced tool for simulating and investigating real vehicular traffic in cities [8]. In the event that the time complexity of the task prevents the usage of traffic simulation framework, we will have to look into developing a mesoscopic traffic simulation model.

**Implement algorithms to optimize traffic flow in simulation:**

We need to implement algorithms capable of improving the fitness of the simulated traffic. In this project, we are going to use genetic algorithms to optimize traffic flow. The details about the function and implementation of the genetic algorithms are given in the following section.

**Experiment with different optimization algorithms:**

We are going to experiment with our proposed approach and other methods of optimizing traffic, to determine the merits of our proposed approach. The details about the experiments are given in the 5th chapter.

**Implement GUI:**

In order to visualize the changes to traffic as the genetic algorithm progressively generates better signal timings, we are going to need a GUI. The GUI is meant to provide a visual representation of the optimization process, as well as to give the user control over the optimization process.

# Chapter 2: Literature Survey

The problem of optimizing traffic flow using traffic light signaling is a complex one and has been researched often. Many methods have been tested in their capability of optimizing traffic. These Methods include reinforcement learning, genetic algorithms, swarm algorithms, neural networks, organic computing and fuzzy logic [3]. For our project, we focused mostly on genetic algorithms. Most of the papers researched in this literature review focus on the application of genetic algorithms and different traffic simulation methods to the traffic optimization problem.

[3]: In this paper, Pawel Gora uses a genetic algorithm to optimize the flow of traffic using traffic simulation framework. The major shortcoming of the approach was the low simulation time of 600 seconds. This was due to the computational complexity of the simulations that were microscopic in nature. Microscopic models are models that continuously or discretely predict the state of individual vehicles [2]. As a result, the improvement in traffic conditions was only minor (3.1%).

[4]: This paper builds upon the work of Pawel Gora. It used a modified version of the traffic simulation framework and a high performance computing cluster to overcome the computational limitations. The results obtained were slightly better than in [3]. However, the results were still not satisfactory, in spite of running much more iterations of algorithms (50 populations). The results obtained after using a mesoscopic traffic simulation were much better. A significant result of this research was the impact of the area simulated and the area used for computing fitness. Performance is improved by expanding area optimized and reducing area for computing fitness.

[5]: This paper also focuses on real time control of traffic by using genetic algorithms to optimize traffic in a microsimulation. The scale of the simulation however, is smaller. The simulation model consisted of only six intersections with one way roads. The results from the prior paper show that optimization will not yield successful results when considering a small area; this is reflected by the fluctuations in the graph of performance and generation number in this paper.

[6]: A graph model is used to represent traffic in this research. They devise a branch and bound algorithm to obtain the optimal solution [6], this method takes a very long time in case of large graphs though. The paper also explores the effects of using other evolutionary algorithms such as genetic algorithms, particle swarm optimization and ant colony optimization. Amongst the tested evolutionary algorithms, genetic algorithms had the best performance.

[7]: In this paper, a genetic algorithm is used for real time optimization. But the algorithm also takes into account the importance of a road in the intersection. The parameter optimized is the total number of cars on a road. The results compared the efficiency of the genetic algorithm in comparison to a fixed timing system. The improvement was of almost 22%. That seems too good, in comparison to prior papers. We suspect that this improvement is due to the different models; similar to the improvement of performance in [4] when a mesoscopic model was used instead of a microscopic one.

[15]: Instead of optimizing traffic using just traffic light signal timings, this paper also examines the impact that optimizing individual car routes can have on overall traffic. Tests were performed to see how route optimization and signal timing optimization affected optimization in general. The results showed that optimizing traffic signaling in conjunction with traffic routes is a promising way to optimize traffic.

[16]: This research applies high performance computing and genetic algorithms to traffic optimization. It also presents an extensive analysis of the impact of different fitness functions on traffic optimization as well as the correlations that exist between the fitness functions. The paper was suggestive of using multi-objective fitness functions to optimize traffic.

The reviewed papers tend to mostly focus on short term control of traffic based on the current traffic conditions. There are two shortcomings of this approach. Firstly, traffic congestions have a tendency to repeat, optimizing based only on current traffic conditions fails to capitalize on this property of traffic. Secondly, actions on traffic have long term consequences that real time control will fail to consider. Real time control forces the use of short term simulations due to computational cost of long term simulations. Hence, we propose a new strategy for traffic optimization that formulates a long term strategy and focuses on short term adaptation to the strategy.

# Chapter 3: Background Knowledge

This chapter elaborates on knowledge that might be needed to understand our project. This chapter is broken down into the following subsections:

* Traffic simulation.
* Traffic Simulation Framework.
* Genetic algorithms.
* Current state of traffic control in Thailand.

## **3.1 Traffic Simulation**

Traffic Simulation is a mathematical model of transportation system that evaluates the patterns of traffic behavior. The model usually accepts census data as an input and generates the estimated behavior of the traffic situation [11].

### 3.1.1 Types of Simulation Models

Traffic simulation models can be classified based on various criteria. They are usually classified based on the following criteria [12]:

* Criteria1: Based on how the elements describing a system change their state, traffic simulation models can be classified as continuous or discrete.
* Criteria2: They can also be categorized by the type of processes represented by the model, into deterministic model and stochastic model.
* Criteria3: The level of detailing is another basis used to classify models. According to this basis, the traffic simulation model is classified into macroscopic model, mesoscopic model, and microscopic model.

**Classification based on criteria1**

* Continuous Model is a model which the state of the system continuously changes as shown in Figure 3.1.
* Discrete Model is a model which the change in the state of system occurs at discrete point of time as shown in Figure 3.2.

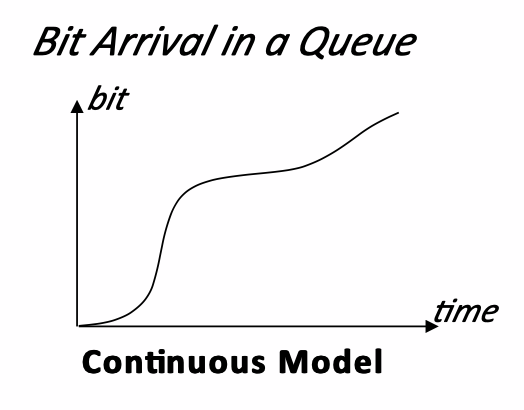
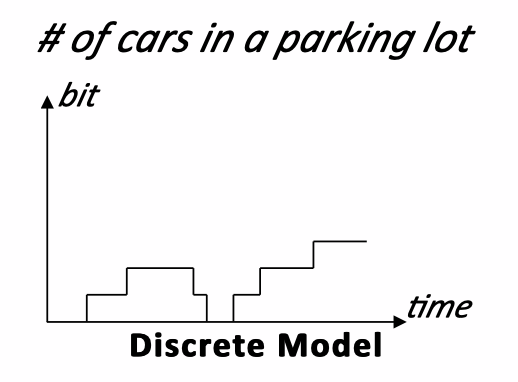
 

Figure 3.1: Continuous Model, taken from [13] Figure 3.2: Discrete Model, taken from [13]

**Classification based on criteria2**

* Deterministic Model represents a fully determined result by non-probabilities or non-random parameters. The model usually represents worst case analysis of the system.
* Stochastic Model includes probability or random parameter. The same set of parameters could lead to dissimilar output.

**Classification based on criteria3**

* Macroscopic Model describes the traffic as overall detail of the intersections. In macroscopic model, three main characteristics those describes the traffic condition are speed, flow, and density. The scope of simulation in macroscopic models is significantly smaller than other types of models; hence the computational complexity of macroscopic models is significantly lower than microscopic models.
* Microscopic Model considers the interactions between vehicles in the stream. In microscopic model, there are many more parameters describing the behavior as compared to macroscopic model, resulting in longer simulations. The data parameters could be flow, density, speed, travel and delay time, long queues, stops, pollution, fuel consumption and shock waves. The level of detail involved also means that microscopic simulation models tend to be more accurate to real life traffic.
* Mesoscopic Model describes traffic information of small groups of vehicles or area. There are two methods of this model which are platoon dispersion and vehicle platoon behavior. Platoon dispersion is a phenomenon where a platoon moves downstream from an upstream intersection, and the vehicles scatters due to the increase of distance occurring by the vehicles speed, its interaction, and other interference. Vehicle dispersion describes a group of vehicles which travels at the same speed and short time headway.

## **3.2 Traffic Simulation Framework**

The Traffic Simulation Framework is an advanced tool for simulating and investigating real vehicular traffic in cities [8]. The TSF model is a cellular automaton-based model inspired by the Nagel-Schreckenberg model for simulating highway traffic.

### 3.2.1 Nagel-Schreckenberg model

The Nagel-Schreckenberg model simulates traffic on a single lane. The road is represented as a tape, divided into cells. At any time, each cell in the model can be occupied or unoccupied by a car. The state can be represented by two variables, the positions of all the cars, and the velocities of all the cars. The velocity of a car is an internal property that can take a limited number of discrete values. The simulation progresses in discrete states, the next state can be obtained from the current state by applying the following rules to all cars at the same time:

* Acceleration: The velocity of the vehicle is advanced by if its velocity is lower than the maximum velocity and if the distance to the next car is larger than .
* Slowing down (due to other cars): If there is a vehicle at site and a vehicle at site and then the vehicle at site reduces its speed to .
* Randomization: The velocity of each vehicle is decreased by randomly based on probability p.
* Car motion: Each car moves forward sites at each step.

The above rules are taken from [10]

### 3.2.2 TSF model

The TSF model inherits the properties of the Nagel-Schreckenberg model and introduces more details in the simulation, to more accurately simulate traffic. The novelties of the TSF model are as follows:

* The road network is a directed graph.
* The position and velocities of cars are not discrete.
* Each car has a pre-selected route calculated using the A\* algorithm based on starting point and destination.
* The road network supports roads with multiple lanes.
* There are different classes of roads, cars on the same road classes behave similarly.
* Every driver has its own profile which affects the behavior of the car.
* Certain crossroads have traffic signals; the traffic signals on the same crossroad are synchronized.
* The vehicle’s slow down before crossroads depending on the route of the vehicle and can also change lanes.
* The movement of the vehicles is similar to that in the last step of the Nagel-Schreckenberg model.

The above details are taken from [8].

### 3.2.3 Traffic Simulation Framework

The Traffic Simulation Framework is an advanced software based on TSF model that can be used to simulate and investigate traffic in cities. The Traffic Simulation Framework can simulate up to 1000000 cars on a road network in real time using a standard desktop machine [8]. As of right now, the Traffic Simulation Framework only provides simulations in the road network of Warsaw. The main functionalities provided by the software are as follows:

* Graphical user interface.
* Traffic simulation with route generation for drivers and modifiable traffic settings.
* Editing distribution of start points and destination points.
* Specifying streets and areas which should be monitored during simulation.
* Showing traffic state during simulation.

The above details are taken from [9].

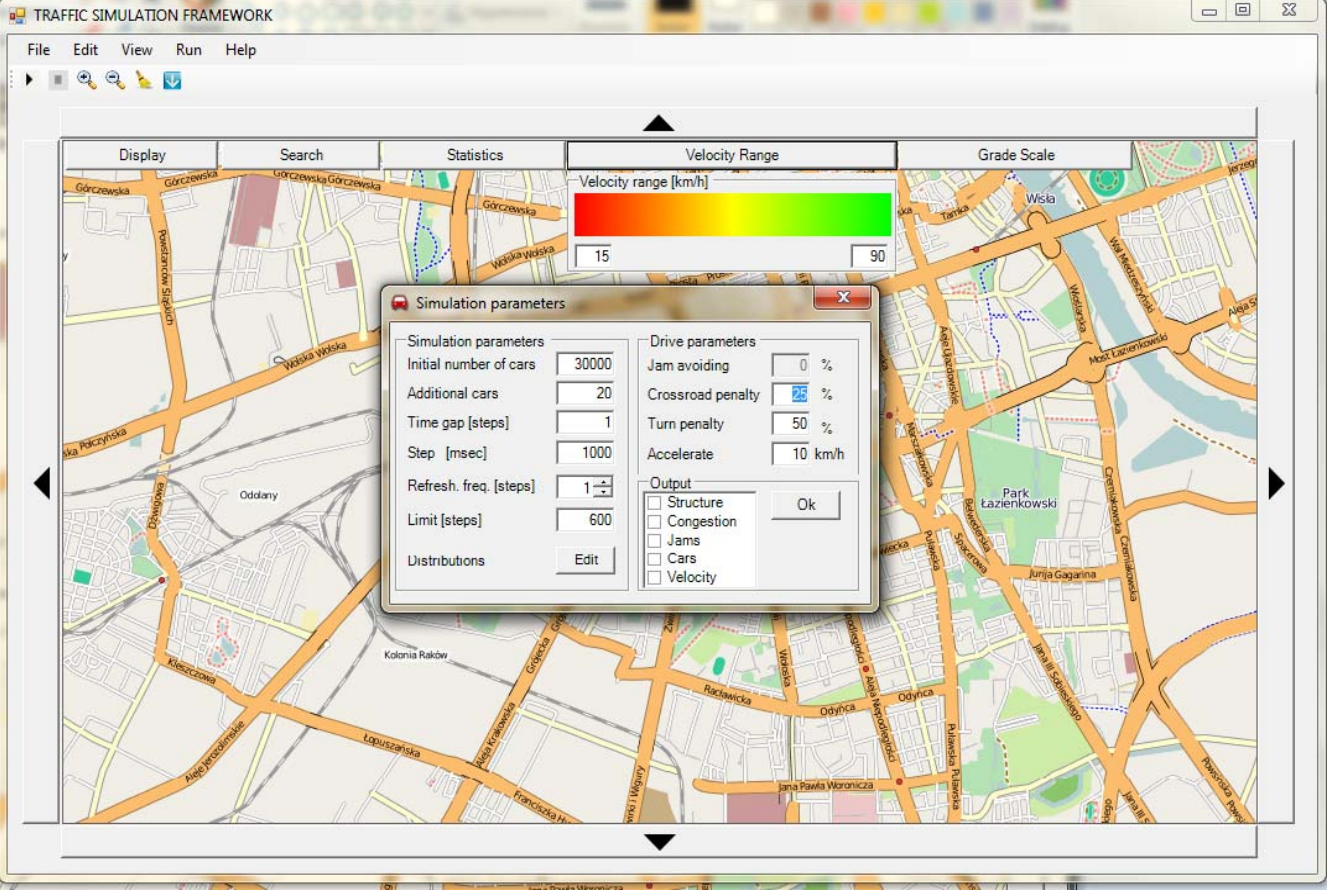


Figure 3.3: Editing simulation parameters and starting simulation, taken from [8]

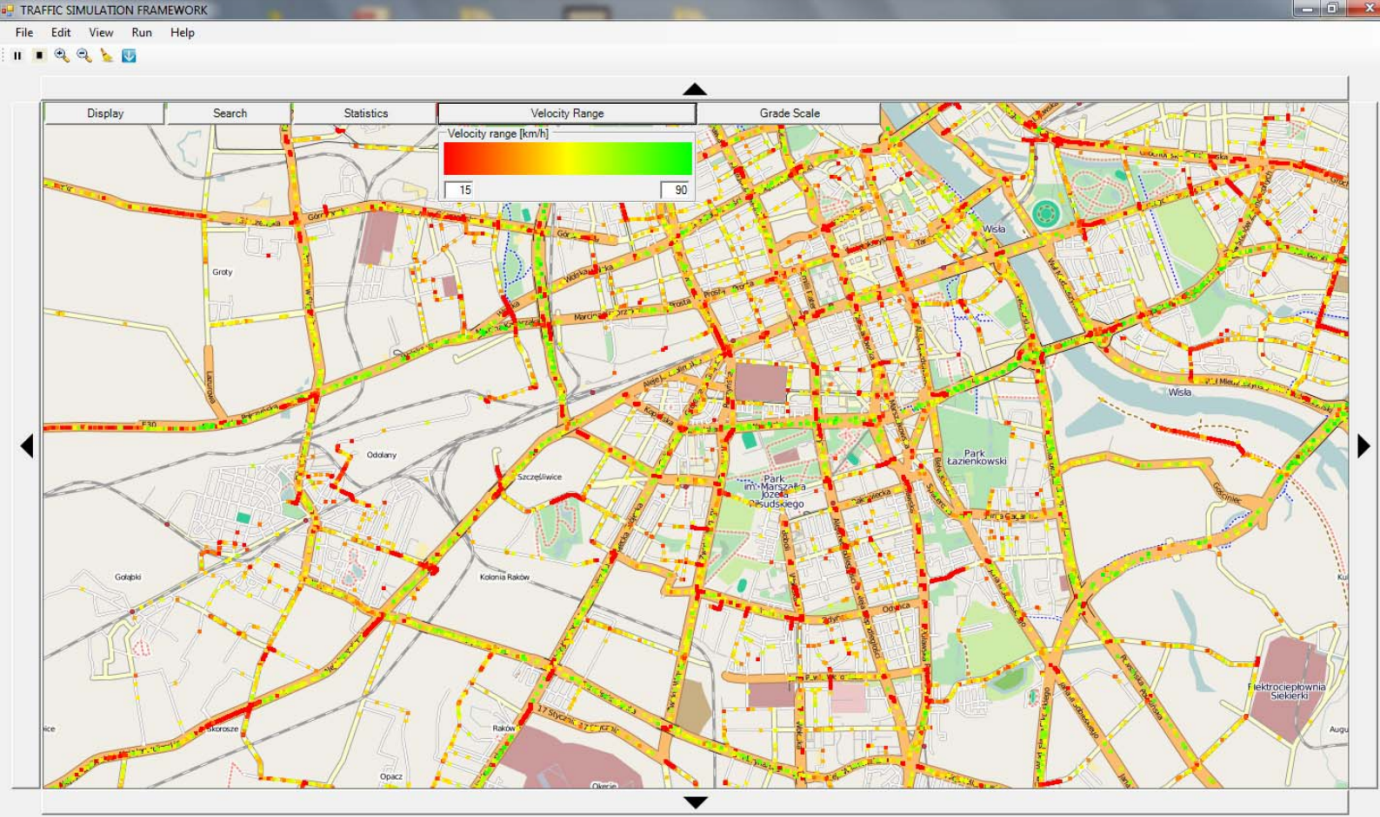


Figure 3.4: The main window of the TSF software, taken from [8]

## **3.3 Genetic Algorithms**

### 3.3.1 Concept

Genetic algorithm is a class of evolutionary algorithms used in computational mathematics to solve optimization and search problems. Evolutionary algorithms were developed based on some phenomena in evolutionary biology, including genetics, mutation, natural selection, and hybridization. For an optimization problem, candidate solutions (called individuals) can be abstractly represented as chromosomes. Traditionally, binary representations (strings of 0 and 1) have been used, but other representations can be used. Evolution begins with a population of completely random individuals out of which the best individuals are used to generate new better populations.

### 3.3.2 Terminology

**Gene**: A gene is a value that is used to represent some characteristic of an individual. For example, if there is a string S=1011, then the four elements 1, 0, 1 and 1 are called the genes. Their values are called the Alleles.

**Chromosome**: Chromosomes can also be called individuals. A chromosome represents a solution and is formed by joining together genes.

**Generation**: A generation refers to a certain population of individuals that has been generated by the genetic algorithm.

**Crossover**: The crossover method is used after selection to breed subsequent generation. There are many different crossover operators that can be used to create new individuals by mixing the genes of the parent individuals.

**Selection**: The goal of selection is to ensure that the best performing (highest fitness) individuals are used to produce the next generation. In selection, a certain number of the best individuals are selected from the population, to be used for subsequent population generation.

**Mutation**: As in biological terms, mutations are used in GA to push the hypothesis to the optimum. Often used with caution, the mutation only flips the bits of the random gene and pushes the entire chromosome toward the offspring, a strategy that evades potential local minima.

**Fitness**: The degree to which each individual adapts to the environment is called fitness. In order to reflect the adaptability of chromosomes, a function that can measure each chromosome in the problem, called the fitness function, is introduced.

### 3.3.4 Computing process

The basic process of genetic algorithms is as follows:

* Initialization: Set the generation counter t=0, and randomly generate M individuals as the initial population P(0).
* Individual evaluation: Calculate the fitness of each individual in the population P(t).
* Selection: apply the selection operator to the group. The purpose of the selection is to obtain the best individuals from a population. The selection operation is based on the fitness assessment of the individual in the group.
* Crossover: The crossover operator is applied to the population. By applying crossover to the selected individuals, we get a new population that includes the descendants of the selected individuals.
* Mutation: The mutation operator changes the gene values at certain locations of an individual’s strings. The population P(t) is subjected to selection, crossover, and mutation operations to obtain the next generation population P(t+1).
* Termination condition judgment: The termination condition can be based on time taken, fitness of best individual or the number of generations. Once the termination condition is met, the individual with the greatest fitness obtained in the evolution process is output as the optimal solution, and the calculation is terminated.

### 3.3.5 Problem domain

Problems that particularly suited to genetic algorithms include optimization and scheduling problems. As a general rule of thumb, genetic algorithms may be useful in problem domains with a complex fitness landscape. The domain of problem solving generally include: Computer-Automated Design, timetable scheduling, neural networks training, optimal design in complex flow fields, etc.

### 3.3.6 Disadvantage

* The coding is not standardized and there is an inaccuracy in the representation of the code.
* A single genetic algorithm encoding cannot fully represent the constraints of the optimization problem. One way to consider constraints is to use thresholds for infeasible solutions, but then, the computation time will increase.
* The genetic algorithm is prone to premature convergence.
* Genetic algorithm has no effective quantitative analysis method for the accuracy, feasibility and computational complexity of the algorithm.

## **3.4 Current State of Traffic Control in Bangkok**

More than 500 traffic lights at the junction in Bangkok have 2 systems which are Automatic Control System and Manual Control System. During the rush hour, the traffic lights are controlled manually by the traffic police [15]. The traffic police estimate the density of the traffic and remotely control the traffic [16]. During regular hour, the Automatic Control System is applied. This kind of system estimates the traffic congestion by collecting the data from the vehicle counting devices and manages the traffic light to balance the congestion for respective roads. This kind of system is applied to 49 junctions in outer Bangkok.

# Chapter 4: Approach

# In this section, we describe our approach for optimizing traffic and the experiments that will be conducted to verify the benefits of our approach.

# Our approach consists of three phases:

* Setting up a traffic simulations that can encapsulate the day to day variation in traffic
* Using a long term genetic algorithm to optimize recurring traffic congestion over a long period of time.
* Use a short term genetic algorithm that uses the signal timings obtained from the long term genetic algorithm to optimize traffic conditions similar to the one optimized by the long term genetic algorithm.

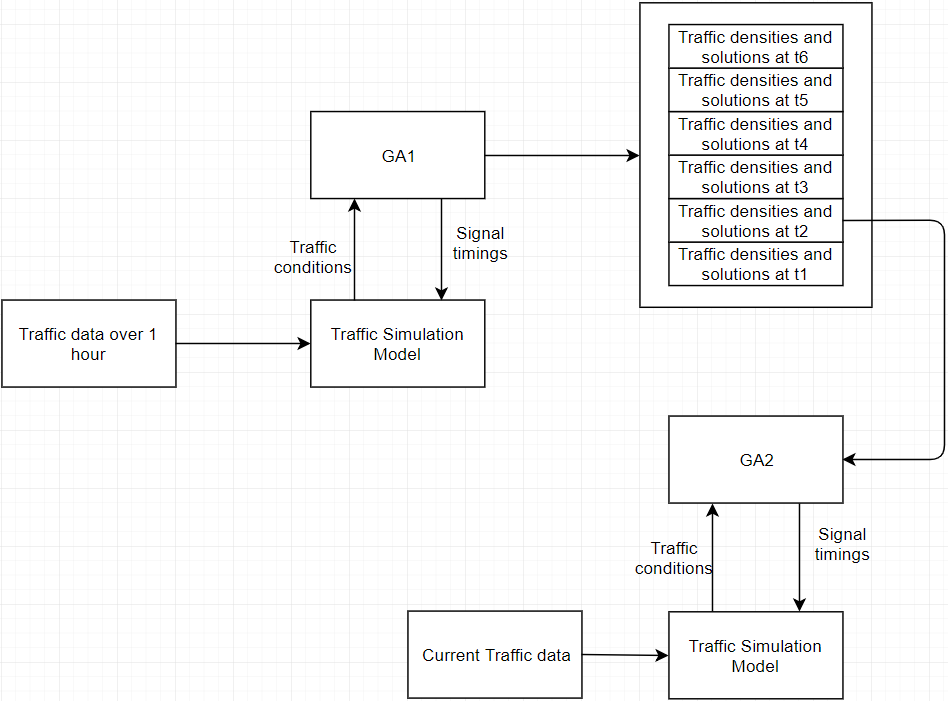


Figure 4.1: Diagram showing the operation of proposed method

## **4.1 Setting up traffic simulation model:**

The traffic simulation model is meant to provide predictions of traffic conditions based on the changing traffic signal timings and changing traffic flow. We are going to use a series of simulations with similar traffic conditions, to test how our hypothesis about using past simulations (GA1) to better improve similar traffic conditions at a later stage (GA2).

In order to set up the model, we need to do the following:

* Select a region to model the traffic
* Setup a model that can accurately predict traffic based on the changing inflow and outflow of traffic from each road.
* Acquire data about the inflow and outflow of traffic for each road during the considered time period. To get information about traffic flow, we need to do the following:
  + Acquire Travel speed data at each road near the intersections.
  + Convert travel speed data into approximate density data.
  + Calculate the fraction of cars exiting into each road from an intersection using the density data
* Use the information gathered about traffic flow at a given time to initialize and direct the traffic simulation.

## **4.2 Optimization 1 with GA1:**

Once we have the traffic data over a time period that relates to repeating traffic congestion, the long term genetic algorithm is meant to optimize a sequence of traffic signals over the entire period. The output of this process is going to be the optimal traffic signal timings at each time step. This task is computationally expensive and will have to be done prior to the real time control task.

Say we have to optimize the traffic over a period *p* with *n* time steps; the figure below shows the optimization process

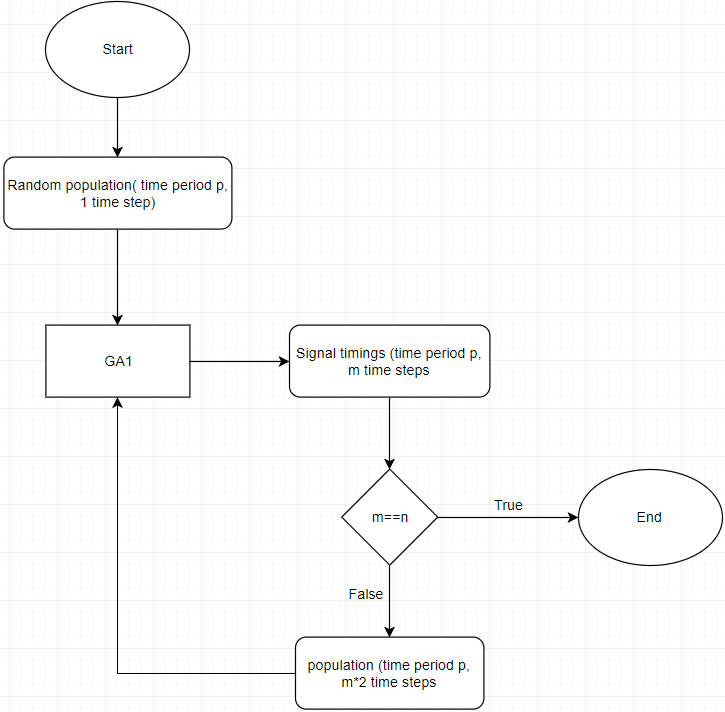


Figure 4.2: Flow diagram for optimization 1

The optimization starts by using GA1 to optimize over period *p* using just one time step. Then the settings from that optimization are used to create a new population. This population is fed to GA1 which now optimizes over period *p* with two time steps, then three time steps and so forth until *n* time steps. GA1 is described in the following section.

**4.2.1 GA1**

**Solution Domain:**

In order to describe the process of obtaining the best solution, we need to define the solution space. The following definitions describe the structure and significance of a genotype for GA1.

**Definition 1:** *Let be the set of traffic lights at a single crossroad.* ***Representant*** *of the set A is any element of the set A. It will be marked as .* ***Representant*** *of any element is . The choice of representant is important, because different representants will yield differing signal timings, however, the eventual result will be the same change in fitness, even though the genotype might look different.*

**Definition 2:** *Let be the set of all crossroads in the road network. Let be the set of* ***representants of all crossroads****.*

**Definition 3:** *Let be the set of* ***possible phase durations*** *for the red and green traffic signals, where correspond to the maximum and minimum phase durations of red and green signals respectively.*

**Definition 4:** *Let be the set of time steps. Let g: G → N be any function mapping the set of representants to the set of possible phase durations and let be a set of different possible mappings from the set of representants to the possible phases.* ***Genotype*** *for the road network is any one to one mapping .*

The above definitions are based on genotype definitions in [3]. A genotype represents the signal phases for each intersection of the road network, throughout the time steps. Initially, the genotypes will be assigned randomly (random phases). Then the genotypes will be used for evolution. A gene in the genotype represents the timing of either the red or green signal at some crossroad at some point in time.

Example: If there were 3 crossroads in the area to be optimized, and the genetic algorithm was considering optimization over two time steps, a single genotype would look like this:

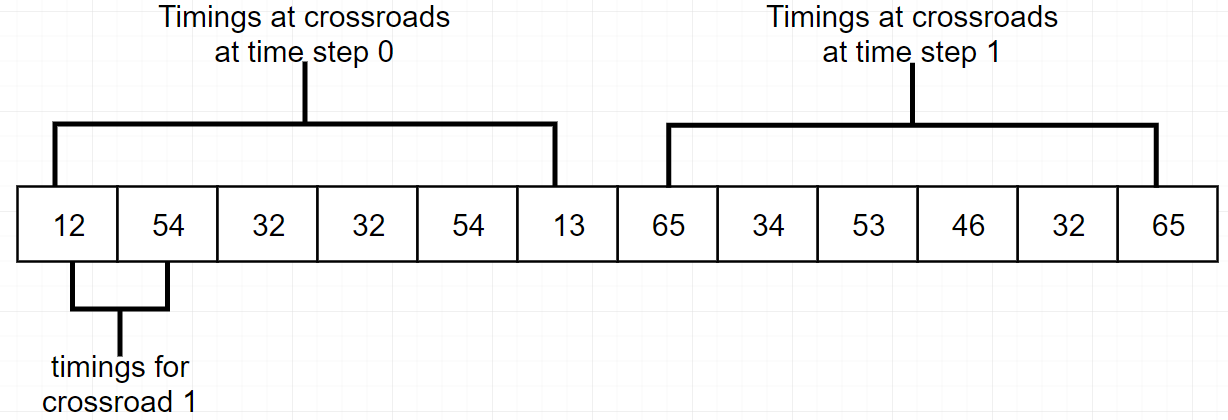


Figure 4.3: Genotype with 3 crossroads and 2-time steps in GA1

**Fitness function**:

For the long term genetic algorithm, we are going to base the fitness of a genotype on either of the following attributes:

* Peak car density at any road in the road network.
* Total waiting time for cars.
* Average travel speed.

We can define and test many fitness functions using the traffic simulation framework; this is not the final list of fitness functions. The primary measure of fitness however is going to be the total waiting time for cars, based on the suggestions of Pawel Gora.

**Selection:**

We have opted to choose the best individuals in the population, due to the small population size imposed by long computation times. Hence, if there are N individuals in the population, we will select of the individuals with the best fitness score.

**Crossover:**

The crossover operator will be selected based on the results of a parameter search. The parameter search is meant to select the suitable hyper parameters for traffic optimization. The results of the search are presented in chapter 6.2.

**Mutation:**

The mutation operator will be selected based on the results of a parameter search. The parameter search is meant to select the suitable hyper parameters for traffic optimization. The results of the search are presented in chapter 6.2.

**Termination:**

We will terminate computation, once a predefined generation limit has been reached.

**Simulation Specifications:**

The specifications of the simulation that the long-term genetic algorithm will try to optimize are as follows:

* Time duration: 20 minutes (we will consider longer durations later)
* Time step duration: 2 minutes
* Time steps (n): (Time duration)/(Time step duration) = 10

**Output of GA1:**

**Definition 5:** *Let D be the set of densities at each road in the network, , where is the density at road k. Let denote the set of road densities at each road in the network at time step t for some genotype. Let* ***DTT*** *be the set of densities at each road in the network at each time step .*

The output includes the traffic signal settings with the highest fitness and the corresponding DTT.

## **4.3 Optimization 2 with GA2:**

The short-term genetic algorithm will be used to achieve real time control of traffic. Instead of initializing the population randomly, GA2 derives its population from the signal settings obtained from GA1. GA2 then evolves these settings further to fit the current traffic conditions. Another modification is that GA2 can try to evolve the signal timings such that the traffic densities mimic the ones obtained from GA1. In that case, the fitness function for this genetic algorithm will be the difference between the simulated traffic densities produced by the individuals in the population and DTT.

**4.3.2 GA2**

**Solution domain:**

In order to describe the process of obtaining the best solution, we need to define the solution space. The following definitions describe the structure and significance of a genotype for GA2.

**Definition 6:** *Let be the set of all crossroads in the road network. Let be the set of representants of all crossroads. Let be the set of possible phase durations for the traffic signals, where correspond to the maximum and minimum durations of red and green phases respectively.* ***Genotype*** *for the road network is any function or.*

The above definition is based on genotype definitions in [3]. A genotype represents the signal phases for each intersection of the road network. Initially, the genotypes will be assigned values based on the evolved genotypes from GA1 corresponding to the same time step. Then the genotypes will be used for evolution. A gene in the genotype represents the timing of either the red or green signal at some crossroad.

Example: If there were 3 crossroads in the area to be optimized, a single genotype would look like this:

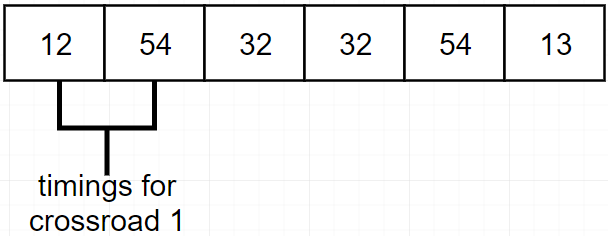


Figure 4.4: Genotype with 3 crossroads in GA2

**Fitness functions**:

In order to specify the fitness of a genotype, we need to define the attribute to be used to measure the fitness. We use the traffic density as the fitness measure. A genotype’s fitness is defined as follows

**Definition 7:** *Consider a genotype G generated by GA2 corresponding to time step i from the long-term simulations conducted prior and its output densities D for each road, , where is the density at road k. The* ***fitness of G*** *is given by the equation , where DTT is the optimal densities at each time step obtained from the long-term genetic algorithm, are the optimal densities at each road at time step i.*

Apart from the above defined measure of fitness, we will also experiment with the fitness functions defined in section 4.2.1

**Selection:**

We have opted to choose the best individuals in the population, due to the small population size imposed by long computation times. Hence, if there are N individuals in the population, we will select of the individuals with the best fitness score.

**Crossover:**

The crossover operator will be selected based on the results of a parameter search. The parameter search is meant to select the suitable hyper parameters for traffic optimization. The results of the search are presented in chapter 6.2.

**Mutation:**

The mutation operator will be selected based on the results of a parameter search. The parameter search is meant to select the suitable hyper parameters for traffic optimization. The results of the search are presented in chapter 6.2.

**Termination:**

We will terminate computation, once a predefined generation limit has been reached.

**Output:**

The output for the short-term genetic algorithm is going to be the traffic signal timings that provide the greatest simulated fitness.

# Chapter 5: Development/Implementation

# In this section, we describe the implementation of the systems required to conduct the experiments.

# This section can be broken down into the following parts:

# Data gathering

# Using traffic data in the simulation

# Genetic algorithms and traffic simulation

# GUI application

# 5.1 Data gathering

# In order to gather the necessary traffic data, to set up the traffic models, we need to do the following:

# Identify and index all the traffic intersections in the selected region.

# Identify and index all the unsegmented road sections (no roads feed in or out from the chosen road).

# Obtain the number of lanes and travel speed data at each road near the traffic intersection that feeds into the road using tomtom traffic API.

# Convert all travel speeds into car densities. We do this by using a simple opencv look up table for now.

# Use the density data to obtain information about what proportion of cars go to which roads at what times.

# In order to get the traffic data, we wrote a function-based script to obtain the travel speeds over an interval of time for a certain set of points using TomTom API. The output of the script is a .csv file which contains the following information: {Timestamp, Speed, Coordinates, Lanes, Length, Road\_id}

# The travel speed data is processed in the next script to convert travel speed data into relative traffic flow information. The input taken is the travel speed data on various roads at various time intervals and information about which roads lead into and out of which intersections. This information is used to calculate the proportion of traffic headed into each road at an intersection at different time intervals. The pseudocode to determine the proportions is as follows:

# def getFlow(roadsIn, roadsOut): consideredLen = 10 flow = {} carsOut = 0 for road in roadsOut: carsOut1 = road.getDensityMoving()\*consideredLen\*road.getLanes() carsOut+=carsOut1 for road in roadsOut: carsOut1 = road.getDensityMoving()\*consideredLen\*road.getLanes() self.flow[road] = carsOut1/carsOut return flow

# def getFlow(roadsIn, roadsOut): consideredLen = 10 flow = {} carsOut = 0 for road in roadsOut: carsOut1 = road.getDensityMoving()\*consideredLen\*road.getLanes() carsOut+=carsOut1 for road in roadsOut: carsOut1 = road.getDensityMoving()\*consideredLen\*road.getLanes() self.flow[road] = carsOut1/carsOut return flow

# 5.2 Using traffic data in traffic simulation

# We utilize the traffic data by generating routes for cars in the traffic simulation. We begin by generating a certain number of cars at the input nodes. The rate of incoming cars at the input nodes will be based on the travel speed data for the corresponding road segment. Each car needs to have a fixed path, leading to some output node, such that overall distribution of cars on roads matches the traffic data obtained. These paths are then supplied to TSF. In order to determine the paths of the cars, the cars at the road segments are sent off into different directions on intersections based on the probability of a car heading into that direction. The probability is based on the traffic flow information determined in the previous section.

# The steps of the algorithm are as follows:

# For each input node, determine a certain number of cars to arrive and the time at which each car arrives

# For each car, determine a route through the city probabilistically (sequentially determine which road to take at each intersection)

# 5.3 Genetic Algorithms and Traffic Simulation

# The genetic algorithms were implemented using the Distributed Evolutionary Algorithms in Python (DEAP) module in python. DEAP provides functions to facilitate crossover, mutation, selection and other actions in genetic algorithms. The structure of the software is as follows:

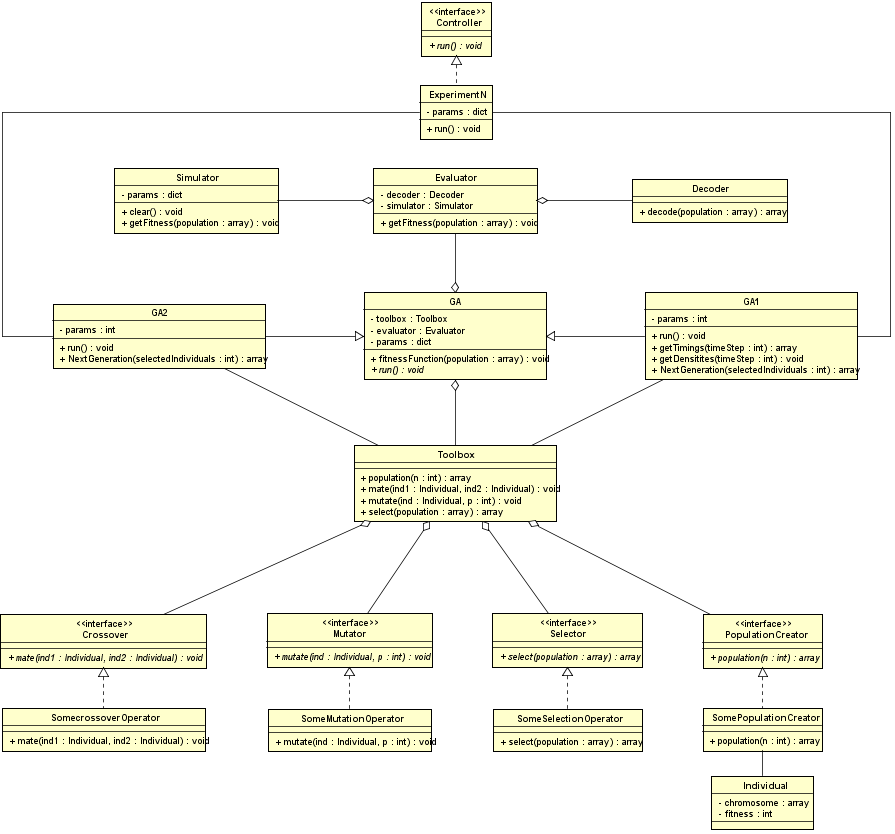


Figure 5.1: class diagram of genetic algorithms

# The sequence of actions required for optimizing traffic is shown as follows:

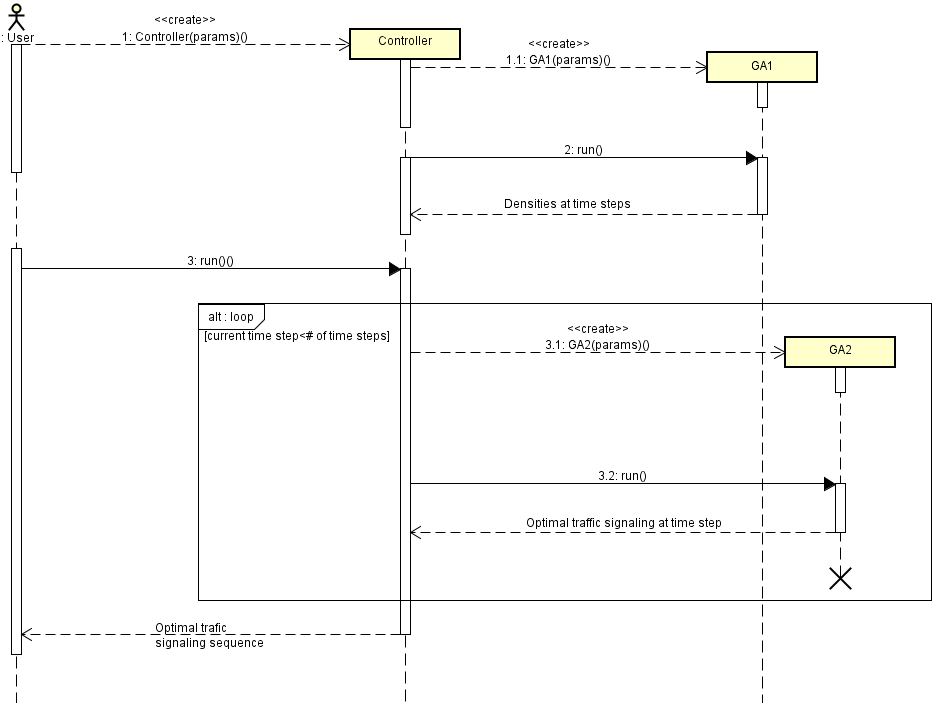


Figure 5.2: Sequence diagram of using genetic algorithms

# The sequence of actions by which the genetic algorithms calculate the fitness of all individuals in a population is as follows:

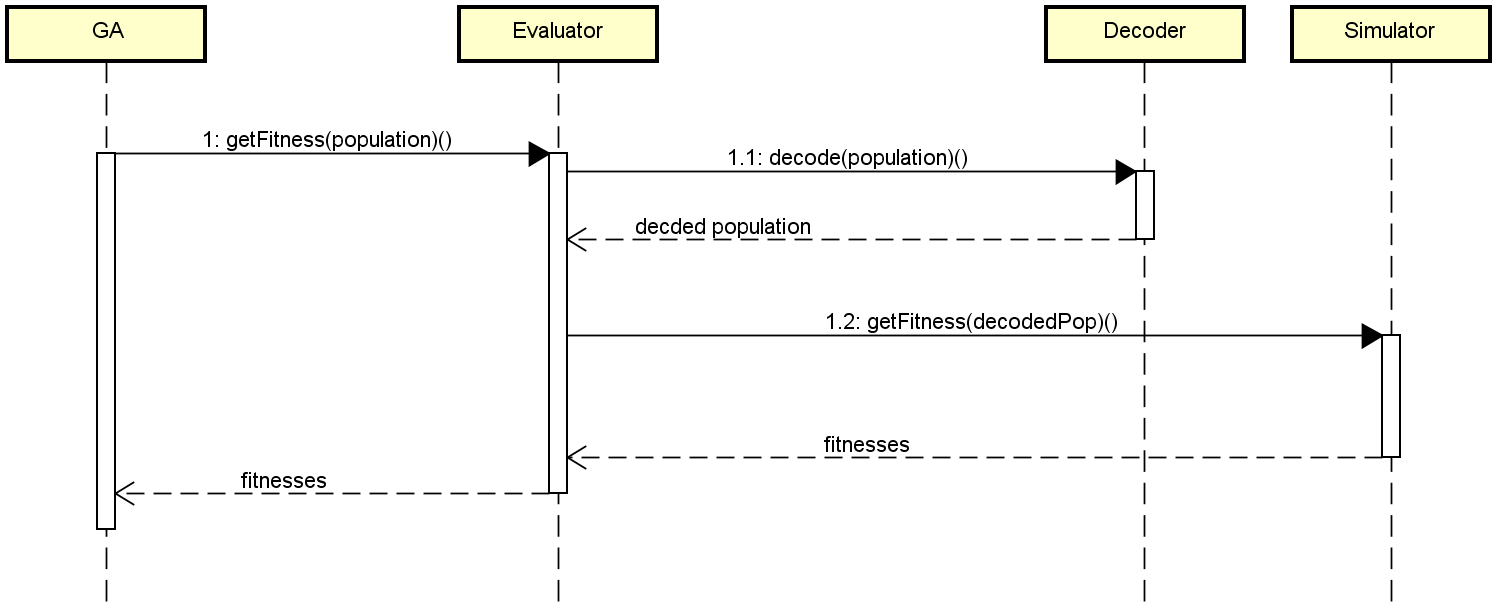


Figure 5.3: Sequence diagram of getting fitness values

The sequence of diagrams to use TSF to obtain the fitness of individuals is as follows:

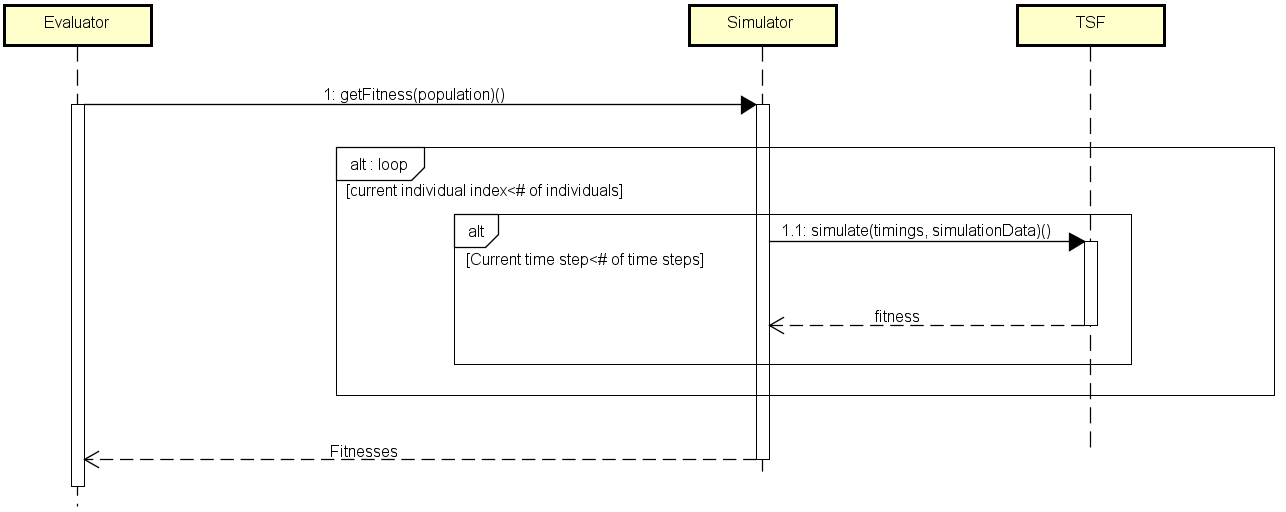


Figure 5.4 Sequence diagram of using TSF

In order to simulate traffic settings, the genetic algorithms use the Simulator object. The simulator object uses the requestStats function to post a request to the Traffic Simulation Framework, which is a microservice in the azure cloud. To determine the fitness of an individual, we send the signal timings at each time step in the individual to be simulated by TSF. TSF responds with the fitness of the timings at each time step. We get the fitness of the individual by summing the fitness of the timings at each time step.

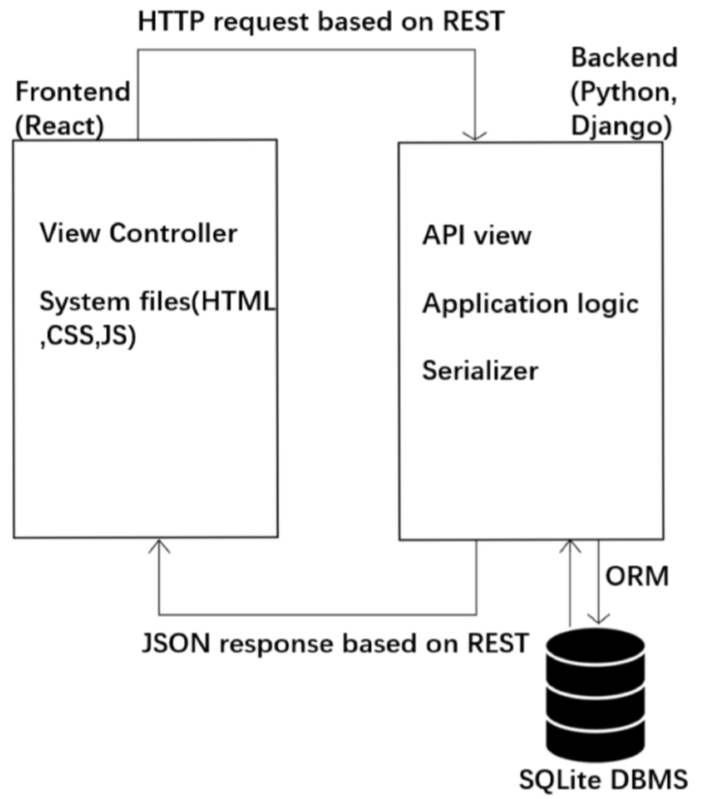
The run time of one experiment can be very long; hence we had to parallelize the requests such that the fitness of multiple individuals could be computed at the same time. We parallelized the requests to the server by using the joblib module in python. A of right now, 16 jobs can be executed in parallel.

## **5.4 GUI application**

The GUI was implemented to give a visual representation of the optimization process, as well as to allow the user to direct the optimization. The application can be opened on any browser compatible with javascript.

**5.4.1 Tools and frameworks used in application**

In this web application, the tools and frameworks used to build the architecture are React (v16.8.4), Python (v3.5), Django (v2.0), SQLite and Representational State Transfer (REST) framework. React is used to build the frontend of the application. Python and Django are used to manage the backend and to communicate with the frontend. REST is used as the communication architecture between frontend and backend. The database of the application is created and maintained using SQLite.



**5.4.2 Use Cases**

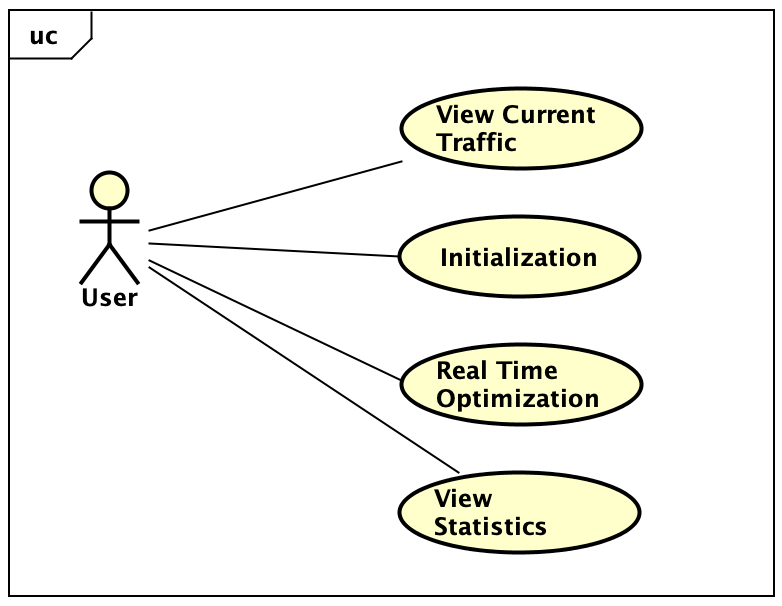


Figure 5.4: Use Case

**Use Case**

The following are the brief use cases for each process

**View Current Traﬃc**: After accessing this page, the user can view and zoom in, zoom out to see the details of the current traﬃc. The default location is Warsaw but they can select other regions to view the traffic.

**Initialization**: The user needs to input GA1 parameters and traffic data in the form of a csv file to start optimization 1. After the user starts the optimization, a loading page will appear to show the progress of the optimization. When the optimization is completed, the user can slide range input to view the traﬃc information of each time step.

**Real Time Optimization**: The user needs to input GA2 parameters and traffic data in the form of a csv file to start optimization 2. After the user starts the optimization, the loading page will appear to show the progress of the optimization. The progress will be updated to show the traffic conditions at the end of the optimization corresponding to each time step.

**View Statistics**: The user can choose to view a statistical report of the previously conducted or currently ongoing optimization.

**5.4.3 Activity Diagram**

# Chapter 6: Experimental Result

# This section describes the experiments conducted and the results obtained from these experiments.

**6.1 Experiment setup:**

The experiments were run using an ASUS ROG G751JT notebook; the following are its specifications:

* Processor: Intel® Core™ i7 4710HQ Processor
* OS: Windows 8.1
* Chipset: Intel® HM87 Express Chipset
* Memory: DDR3L MHz SDRAM, 16GB
* Graphic: NVIDIA® GeForce® GTX970M with 3GB GDDR5

The simulations were run on a 16-core virtual machine on the azure cloud; the settings for the simulations are as follows:

* Number of cars: 30000
* New cars (rate of adding new cars): 20
* Step size: 1000 milliseconds
* Number of steps (simulation duration): 120
* Acceleration: 10
* Crossroad penalty (penalty refers to the deceleration of speed): 0.25
* Turning penalty (penalty refers to the deceleration of speed): 0.5

All the experiments use the same settings unless mentioned otherwise.

**6.2 Tuning Hyperparameters:**

A search was conducted on the hyperparameter space, to determine suitable parameters for GA1 and GA2. The searches for each GA were conducted separately, the difference between the two being the number of time steps. The search was not a complete search due to the time required to conduct such a search. The parameters we were searching for were a mutation operator and a crossover operator.

**6.2.1 GA1:**

The settings used for GA1 when conducting the search are as follows:

* Generations: 3
* Individuals: 40
* Time steps: 3

The results of the grid search are as follows:

|  |  |  |
| --- | --- | --- |
| **Crossover** | **Mutation** | **Improvement** |
| cxOnePoint | mutUniformInt | 3.731442255 |
| cxTwoPoint | mutUniformInt | 6.275004605 |
| cxUniform | mutUniformInt | 1.212009578 |
| cxSimulatedBinaryBounded (eta: 1) | mutUniformInt | 0.902347873 |
| cxSimulatedBinaryBounded (eta: 10) | mutUniformInt | 1.489938493 |
| cxSimulatedBinaryBounded (eta: 100) | mutUniformInt | 5.490394391 |
| cxTwoPoint | mutUniformInt | 5.183908928 |
| cxTwoPoint | mutGaussian | 3.589928480 |
| cxTwoPoint | mutPolynomialBounded (eta: 1) | 2.894928858 |
| cxTwoPoint | mutPolynomialBounded (eta: 10) | 3.984984289 |
| cxTwoPoint | mutPolynomialBounded (eta: 100) | 2.499430957 |
| cxTwoPoint | mutShuffleIndexes | 7.409398180 |

From the above results, we selected cxTwoPoint as the crossover operator and mutShuffleIndexes as the mutation operator.

**6.2.2 GA2:**

The settings used for GA1 when conducting the search are as follows:

* Generations: 3
* Individuals: 40
* Time steps: 1

The results of the grid search are as follows:

|  |  |  |
| --- | --- | --- |
| **Crossover** | **Mutation** | **Improvement** |
| cxOnePoint | mutUniformInt | 8.320903944 |
| cxTwoPoint | mutUniformInt | 10.56302930 |
| cxUniform | mutUniformInt | 4.329009403 |
| cxSimulatedBinaryBounded (eta: 1) | mutUniformInt | 3.493005941 |
| cxSimulatedBinaryBounded (eta: 10) | mutUniformInt | 8.540954045 |
| cxSimulatedBinaryBounded (eta: 100) | mutUniformInt | 10.90430992 |
| cxTwoPoint | mutUniformInt | 14.09403904 |
| cxTwoPoint | mutGaussian | 9.439009930 |
| cxTwoPoint | mutPolynomialBounded (eta: 1) | 4.090399501 |
| cxTwoPoint | mutPolynomialBounded (eta: 10) | 7.765208547 |
| cxTwoPoint | mutPolynomialBounded (eta: 100) | 8.986538774 |
| cxTwoPoint | mutShuffleIndexes | 9.490092953 |

From the above results, we selected cxTwoPoint as the crossover operator and mutUniformInt as the mutation operator.

**6.3 Hypothesis 1:**

Optimization using sequences of timings is better than the default approaches (optimizing over entire time period with 1 sequence of timings or optimizing repeatedly over the time interval with a single sequence of timings)

**Experiment design:**

The performance of each method will be based on the following:

* Fitness after optimization (FAO): The fitness of the system after application of the best solution obtained
* Improvement in fitness of best individual in last generation as compared to best individual first generation. Improvement will not be considered for the last method, because a worst solution doesn’t exist for that method. It consists of multiple optimizations, each with a worst solutions, combining those worst solutions to compute improvement will greatly skew the results
* Time taken to optimize

In the experiments to be conducted, the following are considered to be the major factors:

* Time duration (duration of optimization period)
* Time steps (number of intervals)

The experiments were divided into three sets, one for each method. In the experiments, the duration was varied from 20 min to 60 min, and number of intervals from 4 to 10. The table below shows the experiment parameters for each experiment.

|  |  |
| --- | --- |
| **Experiment** | **Parameters** |
| Experiment 1 | Duration: 20  Intervals: 4 |
| Experiment 2 | Duration: 20  Intervals: 8 |
| Experiment 3 | Duration: 20  Intervals: 10 |
| Experiment 4 | Duration: 40  Intervals: 4 |
| Experiment 5 | Duration: 40  Intervals: 8 |
| Experiment 6 | Duration: 40  Intervals: 10 |
| Experiment 7 | Duration: 60  Intervals: 4 |
| Experiment 8 | Duration: 60  Intervals: 8 |
| Experiment 9 | Duration: 60  Intervals: 10 |

The duration remains the same for all methods in an experiment. The intervals remain the same for method 1 and 3; however, the number of intervals for method 2 is always 1

Other factors that could influence the results were held constant; following are the specifications for those factors:

* Crossover operator: Two-point crossover
* Mutation operator: Shuffle index, mutation probability = 0.1
* Selection operator: Select best
* Number of individuals:
* Number of generation:
* Crossroads: 21
* Maximum phase offset: 119
* Minimum phase offset: 0

The following are the results for experiments relating to hypothesis 1:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment** | **Method 1** | **Method 2** | **Method 3** |
| Experiment 1 | FAO:  Improvement: | FAO:72932  Improvement: 15% | FAO:67251  Improvement: - |
| Experiment 2 | FAO:  Improvement: | FAO: 68427  Improvement: 19% | FAO: 63182  Improvement: - |
| Experiment 3 | FAO:  Improvement: | FAO: 69245  Improvement: 18% | FAO: 59416  Improvement: - |
| Experiment 4 | FAO:  Improvement: | FAO:  Improvement: | FAO:  Improvement: |
| Experiment 5 | FAO:  Improvement: | FAO:  Improvement: | FAO:  Improvement: |
| Experiment 6 | FAO:  Improvement: | FAO: 149479  Improvement: 18% | FAO:  Improvement: |
| Experiment 7 | FAO:  Improvement: | FAO:  Improvement: | FAO:  Improvement: |
| Experiment 8 | FAO:  Improvement: | FAO:  Improvement: | FAO:  Improvement: |
| Experiment 9 | FAO:  Improvement: | FAO:  Improvement: | FAO:  Improvement: |

Note: Method 1 had to be changed; as such the corresponding experiments are incomplete

**6.4 Hypothesis 2:**

Traffic signal settings optimized based on past traffic data can be used to optimize current traffic given that current traffic conditions are similar to traffic data used for optimization

**Experiment design:**

The performance of our proposed method will be based on how well it performs compared to method 2 and 3 in hypothesis 1. The comparison will be based on improvement in simulated traffic and rate of convergence to solution.

The performance of proposed method is going to depend largely on the following factors:

* Time duration
* Time steps
* Variability of traffic: The traffic was varied by randomizing a proportion of the cars’ starting points and destinations. This proportion controls the variability of traffic between past and future optimizations.

The time duration and time steps are fixed based on the best results obtained from prior experiments for hypothesis 1. As such time duration is fixed to … and time steps fixed to …. The traffic variability however is varied from 10% to 50%.

Other factors that could influence the results were held constant; following are the specifications for those factors:

* Crossover operator: Two-point crossover
* Mutation operator: Shuffle index, mutation probability = 0.1
* Selection operator: Select best
* Number of individuals:
* Number of generation:
* Crossroads: 21
* Maximum phase offset: 119
* Minimum phase offset: 0

The following are the results for experiments relating to hypothesis 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment | Traffic variability | FAO | Improvement |
| Experiment 1 | 10% |  |  |
| Experiment 2 | 20% |  |  |
| Experiment 3 | 30% |  |  |
| Experiment 4 | 40% |  |  |
| Experiment 5 | 50% |  |  |

# Chapter 7: Current Progress & Future work

Current progress:

* Finished implementation of genetic algorithms
* Conducted parameter search to tune hyper-parameters of genetic algorithms
* Finished implementing scripts to gather traffic data and generate routes for simulated vehicles based on gathered traffic data
* Parallelized fitness evaluation for individuals in GA to reduce time consumption
* Implemented GUI to visualize the optimization process
* Conducted experiments pertaining to hypothesis 1, discovered problems with the hypothesis and made adjustments to hypothesis, our approach and implementation of genetic algorithms accordingly

Future work:

* Conduct remaining experiments for modified hypothesis 1 and hypothesis 2
* Make the optimization process via GUI modular:
  + Give users the option to supply their own data
  + Allow users to select different simulation frameworks and optimization methods
* Allow saving of state, to prevent loss of progress
* Implement token authentication in the system to improve privacy level and safety level for users
* Design web testing cases to fully test the functionalities of the system
* Integrate GUI with optimization algorithm

# References:

[1] Ratrout, N. T., and Rahman, S. M. 2009. “A COMPARATIVE ANALYSIS OF CURRENTLY USED MICROSCOPIC AND MACROSCOPIC TRAFFIC SIMULATION SOFTWARE”, *The Arabian Journal for Science and Engineering*, vol. 34, no. 1, pp. 121-133.

[2] Boxill, S. A. and Yu, L. 2000. “An Evaluation of Traffic Simulation Models for Supporting ITS Development”, Center for Transportation Training and Research Texas Southern University, Houston, Texas.

[3] Gora, P. 2011 ) “A genetic algorithm approach to optimization of vehicular traffic in cities by means of configuring traffic signals”, *Emergent Intelligent Technologies in the Industry*, pp. 1-10, ISBN: 978-3-642-22731-8.

[4] Gora, P. and Pardel, P. W. 2015. "Application of Genetic Algorithms and High-Performance Computing to the Traffic Signal Setting Problem”.

[5] Kwasnicka, H. and Stanek, M. 2006. “Genetic Approach to Optimize Traffic Flow by Timing Plan Manipulation”. *ISDA 2006 Proceedings*, vol. II, pp. 1171–1176.

[6] Chen, S.W., Yang, C.B. and Peng, Y.H. 2007. “Algorithms for the Traffic Light Setting Problem on the Graph Model”. *the 12th Conference on Artificial Intelligence and Applications*, TAAI.

[7] Singh, L., Tripathi, S. and Arora, H. 2009. "Time Optimization for Traffic Signal Control Using Genetic Algorithm", *Int. J. of Recent Trends in Engineering and Technology*, vol. 2, no. 2.

[8] Gora, P. 2012. “Traffic Simulation Framework”. *14th International Conference on Modelling and Simulation*, pp. 345-349, ISBN: 978-0-7695-4682-7

[9] Gora, P. 2009. “Traffic Simulation Framework – a cellular automaton based tool for simulating and investigating real city traffic”. *Recent Advances in Intelligent Information Systems*, pp. 642-653.

[10] Nagel, K. and Schreckenberg, M. 1992. “A cellular automaton model for freeway traffic”. *Journal de Physique 2 (1992)*, pp. 2221-2229.

[11] J. D. Leonard II, 2001. "An Overview of Simulation Models in Transportation", [Online]. Available: https://web.archive.org/web/20081101205621/http://www.sisostds.org/webletter/siso/iss\_79/art\_429.htm. [Accessed: 08- Nov- 2018].

[12] Mathew, T. 2014. "Microscopic Traffic Simulation", IIT Bombay.

[13] Reiter, J. 2015. "Application of a Markov chain traffic model to the Greater Philadelphia Region", https://www.slideshare.net/JosephReiter1/traffic-model.

[14] Azlan N.N., and Rohani, M.M. 2018. "Overview Of Application Of Traffic Simulation Model". *MATEC Web of Conferences*, vol. 150, p. 03006.

[15] Dezain, H., Marranhghello, N. and Damian, F. 2014. “Genetic algorithm-based traffic lights timing optimization and routes definition using Petri net model of urban traffic flow”. *IFAC Proceedings Volumnes*, vol. 47, issue 3, pp 11326-11331

# [16] Javier, J., Sanchez-Medina, Manuel J. and Enrique, R. 2010. “Traffic Signal Optimization in “La Almozara” District in Saragossa Under Congestion Conditions, Using Genetic Algorithms, Traffic Microsimulation, and Cluster Computing”. *IEEE Transactions on Intelligent Transportation Systems,* vol. 11, issue 1, pp 132-141