Optimal Traffic Light Signaling Based on Genetic Algorithm Approach

Alok Yadav
Tan Tianyu
Sajeerat Aussavaruengsuwat

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2018
KMITL-2019-IC-B-003-010

Thesis - Academic year 2018

Bachelor of Engineer in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

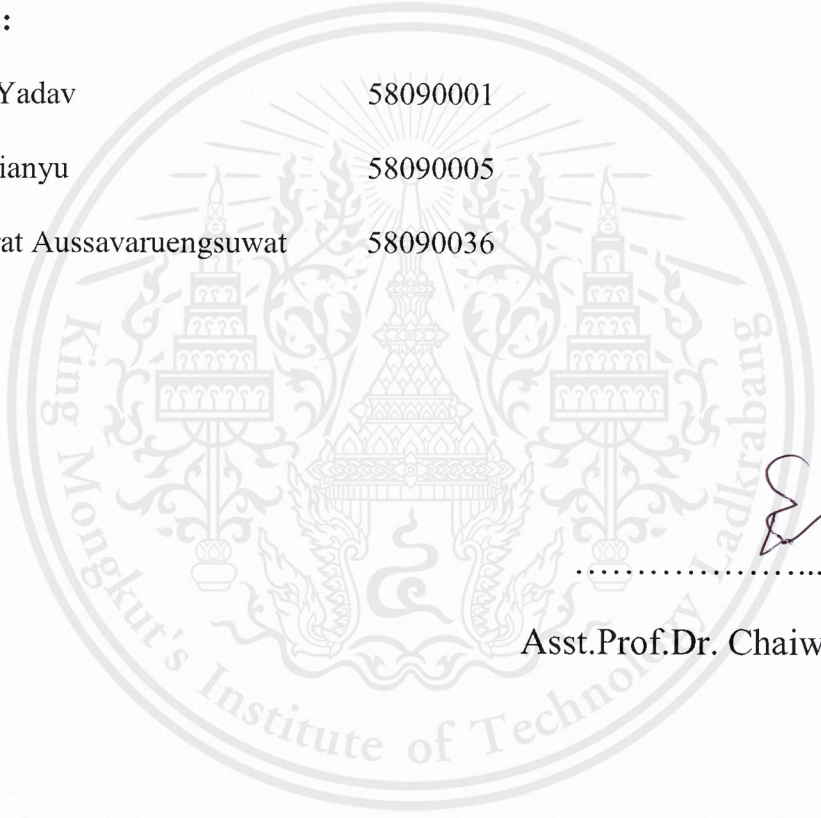
Title – Optimal Traffic Light Signaling Based on Genetic Algorithm Approach

Authors:

1. Alok Yadav 58090001

2. Tan Tianyu 58090005

3. Sajeerat Aussavaruengsuwat 58090036



.....

Asst.Prof.Dr. Chaiwat Nuthong

Advisor

Date: 27/6/2019

Acknowledgement

This project was a collaborative effort between many people. We are grateful to all people who provided assistance in the completion of the project.

We would like to thank Dr. Chaiwat Nuthong for advising our project. Advice given by Dr. Chaiwat Nuthong was very helpful in designing appropriate experiments and developing our algorithms. We appreciate his guidance and patience.

We would also like to express our gratitude to Dr. Pawel Gora, who allowed us to use Traffic Simulation Framework for our simulations and for lending us the computational resources necessary to carry out our experiments.

Abstract

This thesis presents and tests different approaches for optimizing traffic signal timings using genetic algorithms in order to optimize traffic flow. Three approaches are designed and tested, which differ in how they handle traffic variability. Two of the approaches exist in past literature; the last approach is the novelty of our research.

The approaches are implemented with the help of Distributed Evolutionary Algorithms Python (DEAP) module for python and Traffic Simulation Framework, provide by Dr. Pawel Gora. The approaches are tested by varying simulation parameters, such as simulation time and optimization parameters such as number of time intervals. Performance is measured in terms of total waiting time for simulated cars. The proposed approach failed to perform satisfactorily due to a large search space resulting in poor convergence of the genetic algorithm.

Table of Contents

List of Figures

List of Tables

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 many cities, 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 time series 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.

Thesis Structure

This thesis consists of seven chapters which are arranged as follows:

- Objectives and problem description: Defines the problem that we are trying to deal with and establishes the objectives and scope of the research
- Related works: Examines several other publications related to optimization of traffic signal timings
- Background knowledge: Covers topics necessary in order to understand the thesis
- Approach: Covers different approaches that were experimented with to optimize traffic flow
- Implementation: Describes the implementation of the components necessary to carry out the required experiments and appropriately display the results
- Experimental results: Displays the results obtained from our experiments
- Conclusion: Discusses the results and concludes the thesis

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:

1. **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.

2. **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.

3. **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 chapter 4.

4. **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.

5. **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.

1.4 Scope

The following defines the scope of the research:

- Develop GA based algorithms capable of determining traffic signal timings that reduce waiting time for cars in a simulation
- Different algorithms will be compared in their ability to reduce waiting time with changing experiment parameters
- Traffic Simulation Framework will be used to simulate traffic, limiting the area of simulation to Warsaw and optimization period to 1 hour

Chapter 2
Related Works

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.

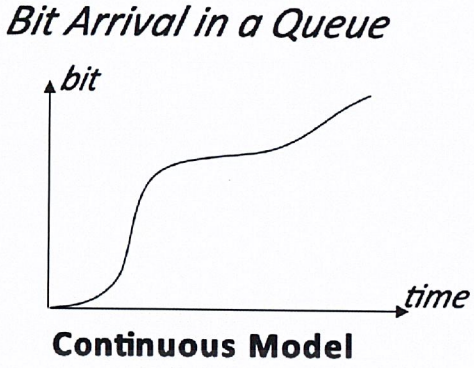


Figure 1: Continuous Model, taken from [13]

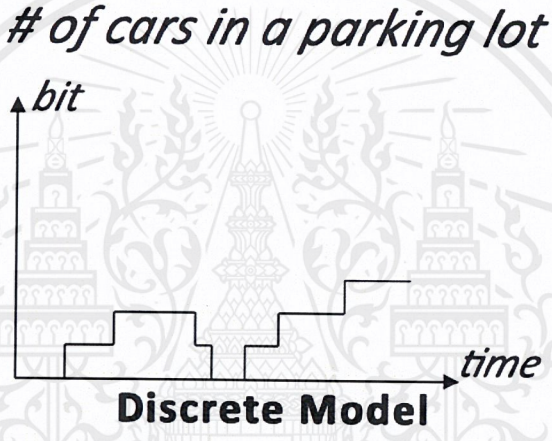


Figure 2: Discrete Model, taken from [13]

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 $\mathbf{v}$ is advanced by $[\mathbf{v} \rightarrow \mathbf{v}+\mathbf{1})$ if its velocity is lower than the maximum velocity and if the distance to the next car is larger than $\mathbf{v}+\mathbf{1}$.
- Slowing down (due to other cars): If there is a vehicle at site $\mathbf{i}$ and a vehicle at site $\mathbf{i}+\mathbf{j}$ and $\mathbf{j} \leq \mathbf{v}$ then the vehicle at site $\mathbf{i}$ reduces its speed to $\mathbf{j}-\mathbf{1}[\mathbf{v} \rightarrow \mathbf{j}-\mathbf{1}]$.
- Randomization: The velocity of each vehicle is decreased by $[\mathbf{v} \rightarrow \mathbf{v}-\mathbf{1})$ randomly based on probability p.
- Car motion: Each car moves forward $\mathbf{v}$ 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].

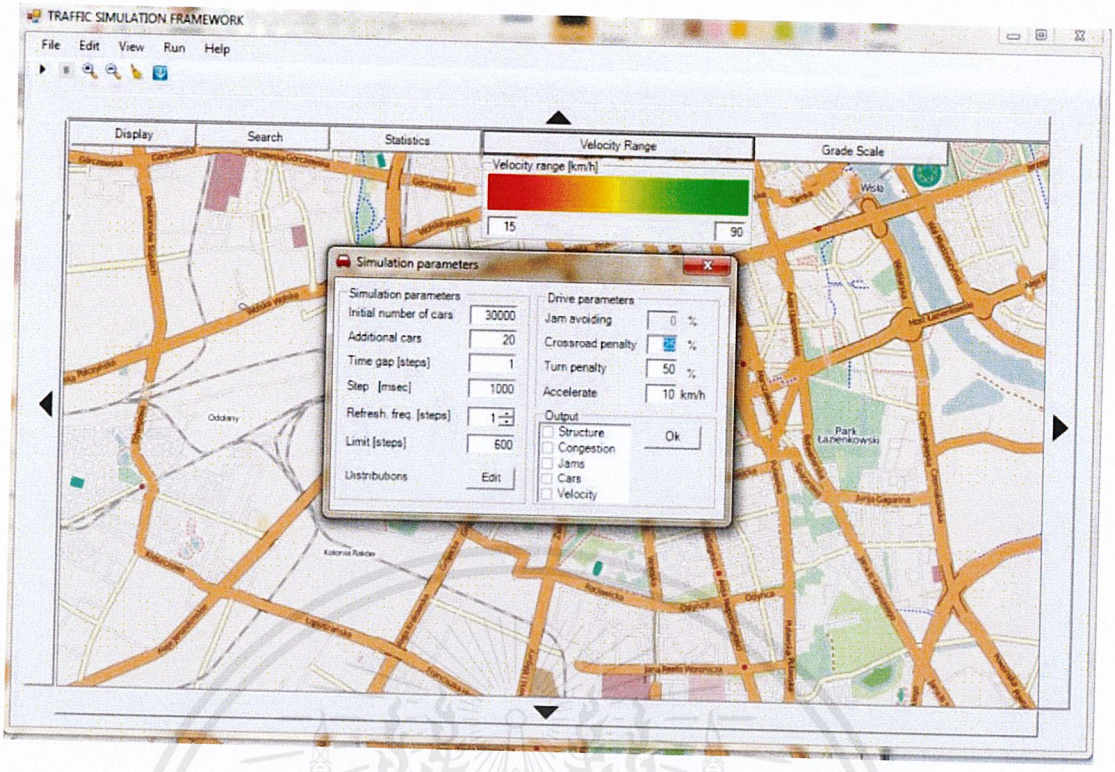


Figure 3: Editing simulation parameters and starting simulation, taken from [8]

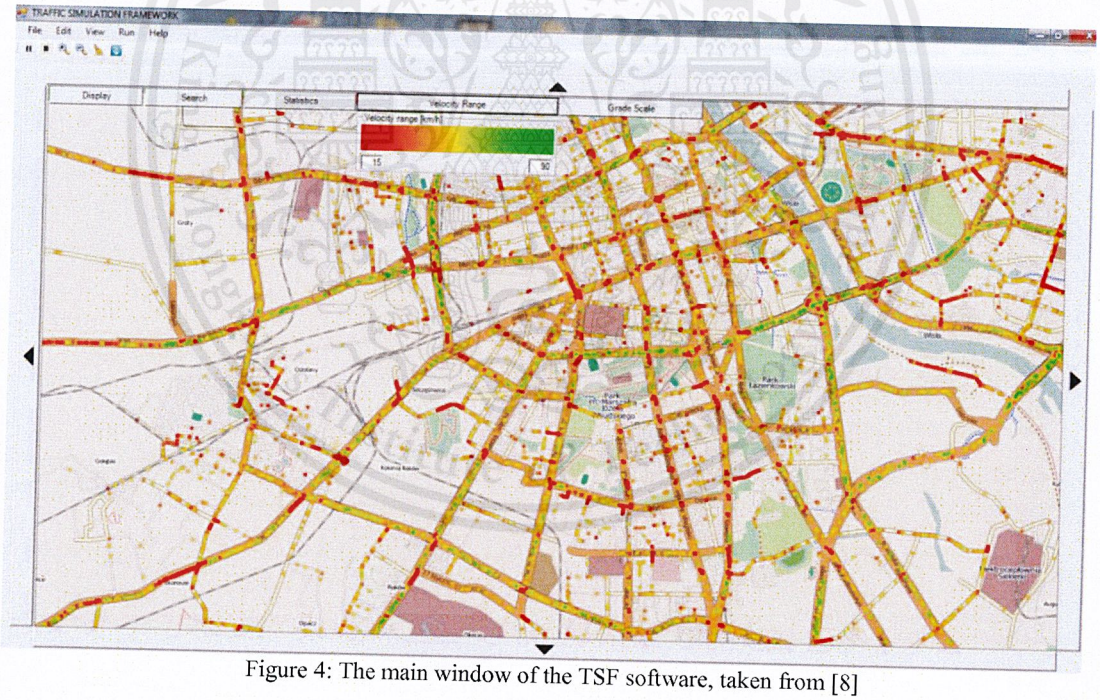


Figure 4: The main window of the TSF software, taken from [8]

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: In GA, a new hypothesis collection consists of previous assumptions, by selecting the complete chromosome (usually has a high adaptability) in order to move forward to a new generation of unspoiled (choice), by turning the existing complete chromosome and move it forward to a new generation (mutations), or the most common is, through the use of existing genes as the parent children cultivate a new generation of chromosomes.

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).
- 9
- 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 generation. 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 seem 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 complex adaptive patterns, namely, mutations associated with crossing, designed to keep populations away from local optimization, and traditional hill-climbing algorithms may be stuck observing that commonly used crossing operators cannot alter any uniform population. A single mutation provides the ergodicity of the entire genetic algorithm process (seen as a markov chain).

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.

Chapter 4

Approach

In order to use genetic algorithms (GA) and traffic simulations to optimize traffic, we need to define a genotype for the GA.

Definition 1: Let $A = \{A_1, A_2, A_3, \dots A_k\}$ 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 $r(A)$. **Representant** of any element $A_i \in A$ is $r(A)$: $\forall A_i \in A_r(A_i) = r(A)$. 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 $C = \{C_1, C_2, C_3, \dots, C_n\}$ be the set of all crossroads in the road network. Let $G = \{r(C_1), r(C_2), r(C_3), \dots, r(C_n)\}$ be the set of **representants of all crossroads**.

Definition 3: Let $N = \{min_p, \dots, max_p\}$ be the set of **possible phase offsets** for the red and green phase durations (difference between durations of the two phases.

Definition 4: Let $C = \{C_1, C_2, C_3, \dots, C_n\}$ be the set of all crossroads in the road network. Let $G = \{r(C_1), r(C_2), r(C_3), \dots, r(C_n)\}$ be the set of representants of all crossroads. Let $N = \{min_p, \dots, max_p\}$ be the set of possible phase offsets for traffic signals (difference between green phase and red phase duration). **Genotype** for the road network is any function Genotype: $G \rightarrow N$ or.

The above definitions are based on genotype definitions in [3]. A genotype represents the signal phases for each intersection of the road network.

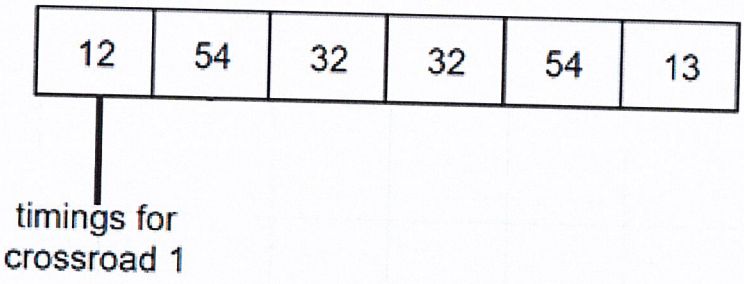


Figure 5: Genotype for 6 crossroads in GA

The traffic optimization process goes as follows:

- A traffic simulator is set up to test the genotypes generated by the GA
- GA generates a population of random genotypes, each genotype representing the timings at traffic signals across the road network for the considered time duration (optimization period)
- The fitness of these genotypes is calculated by using the traffic simulator, to measure how well each genotype is able to handle traffic
- The GA selects some of the genotypes based on some criteria and uses them to generate the next population of genotypes that are then again tested using the traffic simulator.

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 Genetic Algorithm Specifications:

Initialization:

Depends on approach used

Fitness function:

Depends on approach used

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 $\sqrt{N}/2$ 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.

4.3 Approach 1:

In this approach, the genotypes represent the traffic signal timings for the entire optimization period. The approach is identical to the one explained previously, at the beginning of this chapter.

GA Initialization:

The genotypes will be initialized randomly.

GA fitness function:

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

4.4 Approach 2:

Let **N** represent the duration of the optimization period. In this approach, the optimization period is divided into **n** smaller intervals, each of which will be called a time step. Then we carry out **n** optimizations for each of the smaller intervals, each optimization identical to the optimization in approach 1.

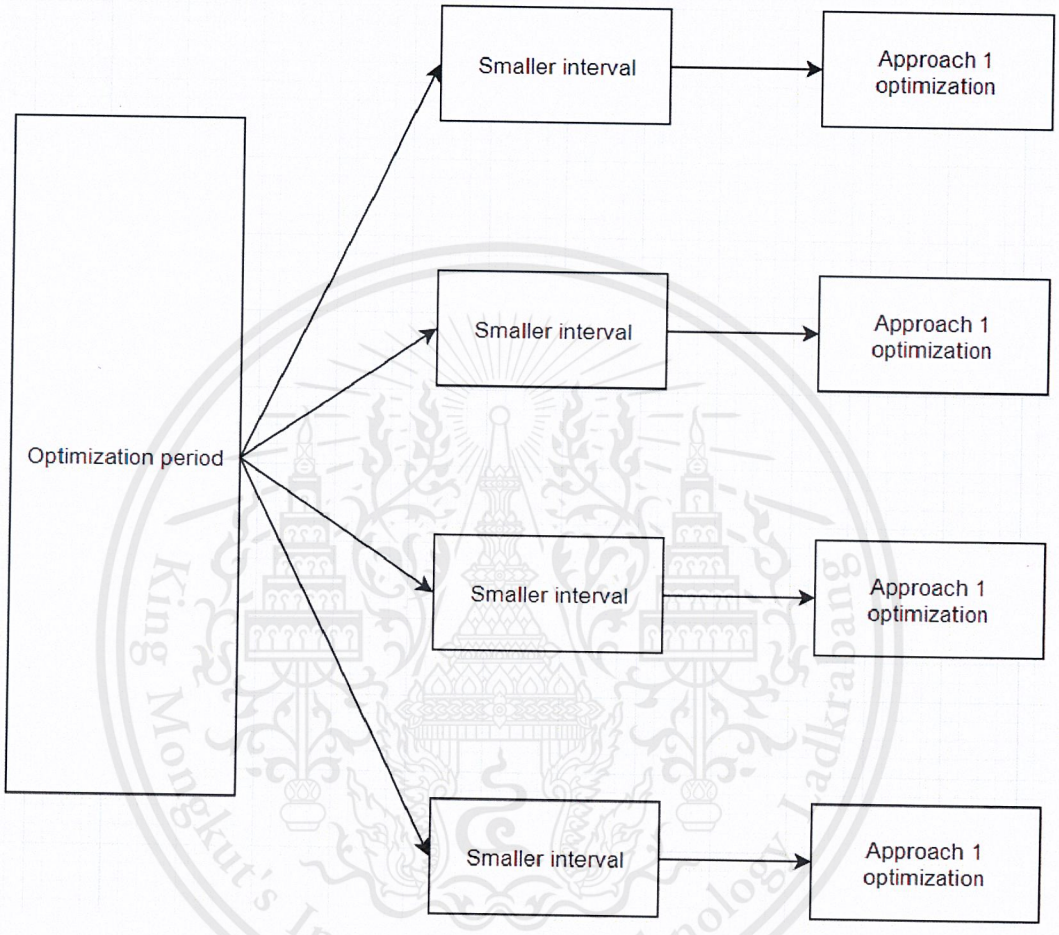


Figure 6: Approach 2

GA Initialization:

The genotypes will be initialized randomly.

GA fitness function:

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

4.5 Approach 3:

Let **N** represent the duration of the optimization period. In this approach, the optimization period is divided into **n** smaller intervals, each of which will be called a time step. Then, one optimization is carried out, in which each genotype is also segmented into **n** sub-genotypes. Each sub-genotype represents the timings in one of the smaller intervals. The genotype representation in approach 3 is as follows:

Definition 4: Let $T = \{t_1, t_2, t_3, \dots, t_{num\_steps}\}$ be the set of time steps. Let g: $G \rightarrow N$ be any function mapping the set of representants to the set of possible phase durations and let $GN = \{g_1, g_2, g, \dots, g_{num\_steps}\}$ 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 Genotype: $GN \rightarrow T$.

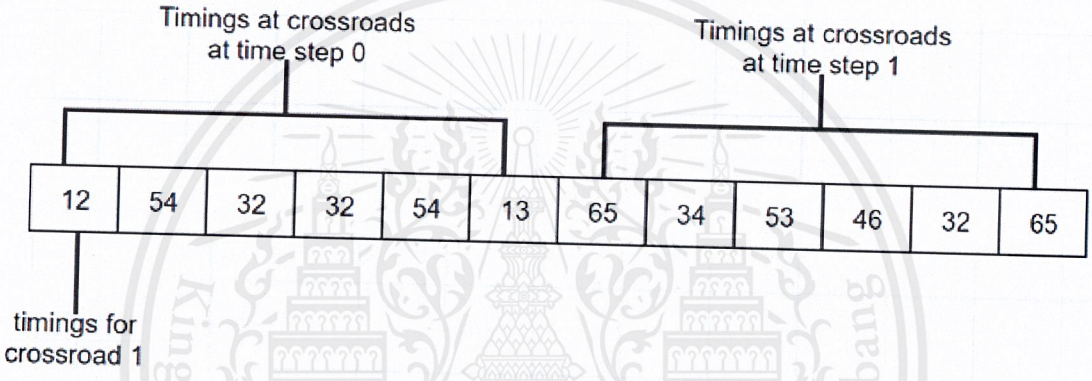


Figure 7: Genotype with 6 crossroads and 2 time steps in approach 3

Optimizing traffic by this approach can be very time consuming. Using this approach in real-time optimization is hence not practical. In order to circumvent this issue, we propose the following optimization procedure:

- Use above approach to optimize recurring traffic congestion over a long period of time based on historic traffic data.
- Use another genetic algorithm that uses the signal timings and traffic densities obtained from the prior optimization to optimize traffic conditions similar to the one optimized previously. This optimization can be done in real time, since it is similar to approach 2, the difference being the fitness function (density based) and the initialization (non-random, based on population from prior optimization)

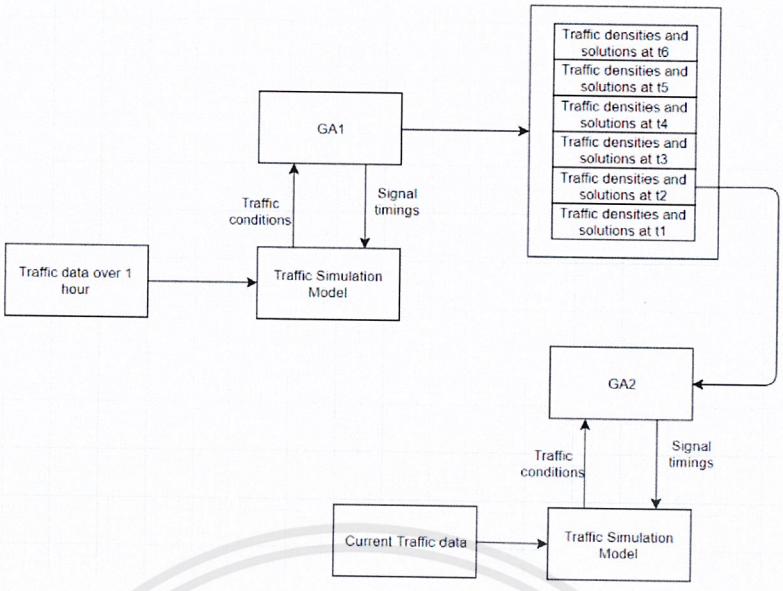


Figure 8: Diagram showing the operation of proposed method

4.5.1 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.

Initialization:

The genotypes will be initialized randomly.

Fitness function:

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.

Output of GA1:

Definition 5: Let D be the set of densities at each road in the network, $D = \{D_1, D_2, D_3, \dots, D_n\}$, where D_k is the density at road k. Let D^t 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 $DTT = \{D^1, D^2, D^3, \dots, D^{num\_steps}\}$.

The output includes the genotypes with the highest finesses and their corresponding DTTs.

4.5.2 Optimization 2 with GA2:

The second 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.

Initialization:

Optimization 1 was done over time period N split into **n** smaller intervals. Optimization 2 will be done similar to approach 2, in that there will be **n** repeated optimizations. However, the initialization of the genotypes won't be random. For optimization **k** out of the **n** optimizations, the population will be generated based on the **k**th regions of the selected genotypes from optimization 1. The selected regions will be treated as individual genotypes and be crossed with each other to create the population.

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 can be defined as follows

Definition 6: 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, $D = \{D_1, D_2, D_3, \dots, D_n\}$, where D_k is the density at road k. The **fitness of G** is given by the equation $f = ||DTT[i] - D||$, where DTT is the optimal densities at each time step obtained from the long-term genetic algorithm, DTT[i] are the optimal densities at each road at time step i.

Apart from the above defined measure of fitness, we will also experiment with other fitness functions listed as follows:

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

Output of GA2:

The output for GA2 is going to be the traffic signal timings that provide the greatest simulated fitness.

Chapter 5

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
```

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:

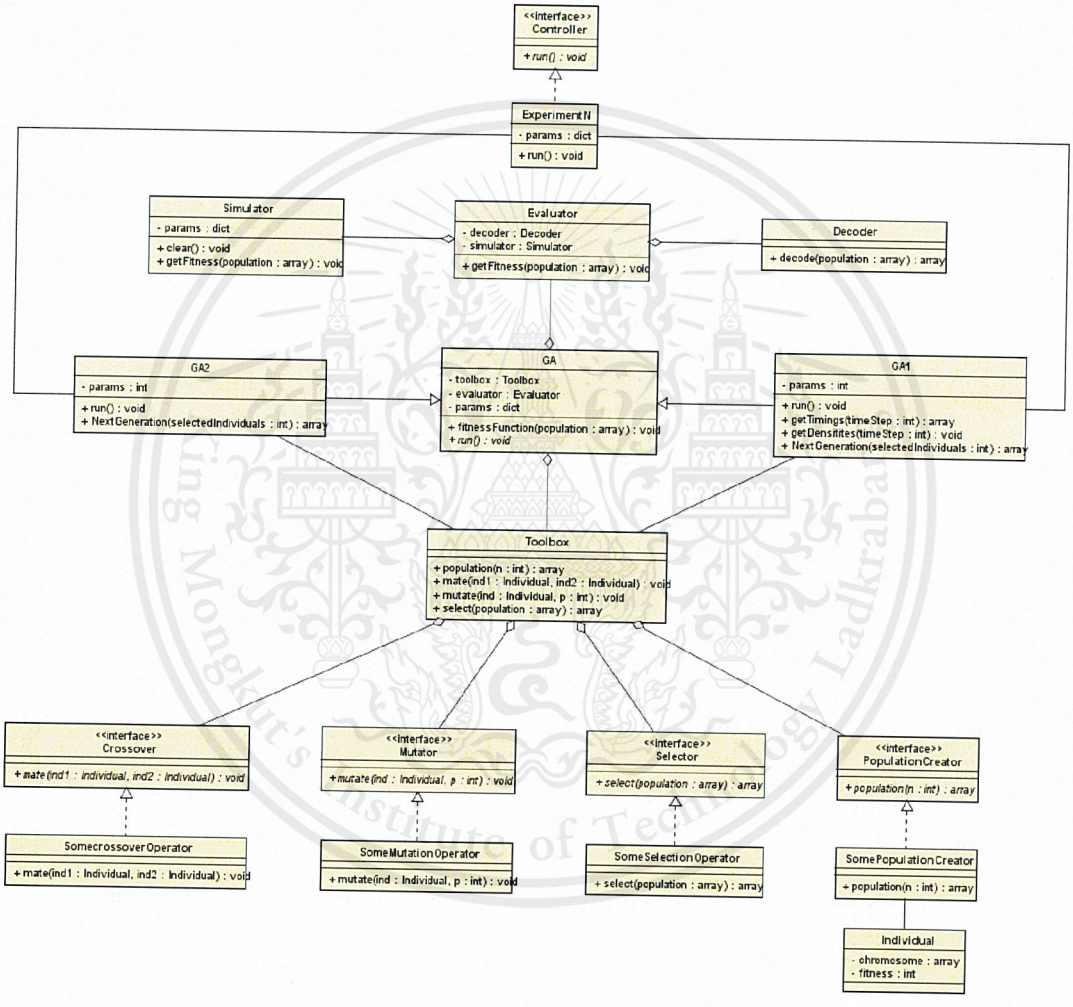


Figure 9: Class diagram of genetic algorithms

The sequence of actions required for optimizing traffic by approach 1 is shown as follows:

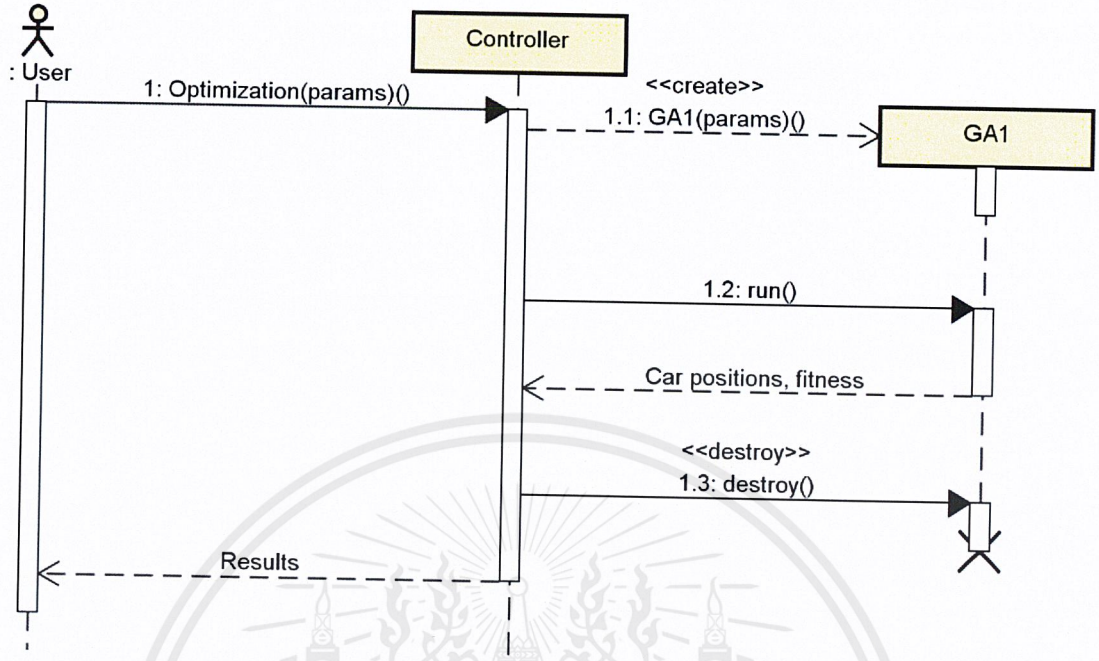


Figure 10: The sequence diagram for optimizing traffic by approach 1

The controller is one of the experiment files, or the UI application. For approach 1, we use GA1 implemented for approach 3, and set the parameters such that the optimization period doesn't get split into smaller intervals.

The sequence of actions required for optimizing traffic by approach 2 is shown as follows:

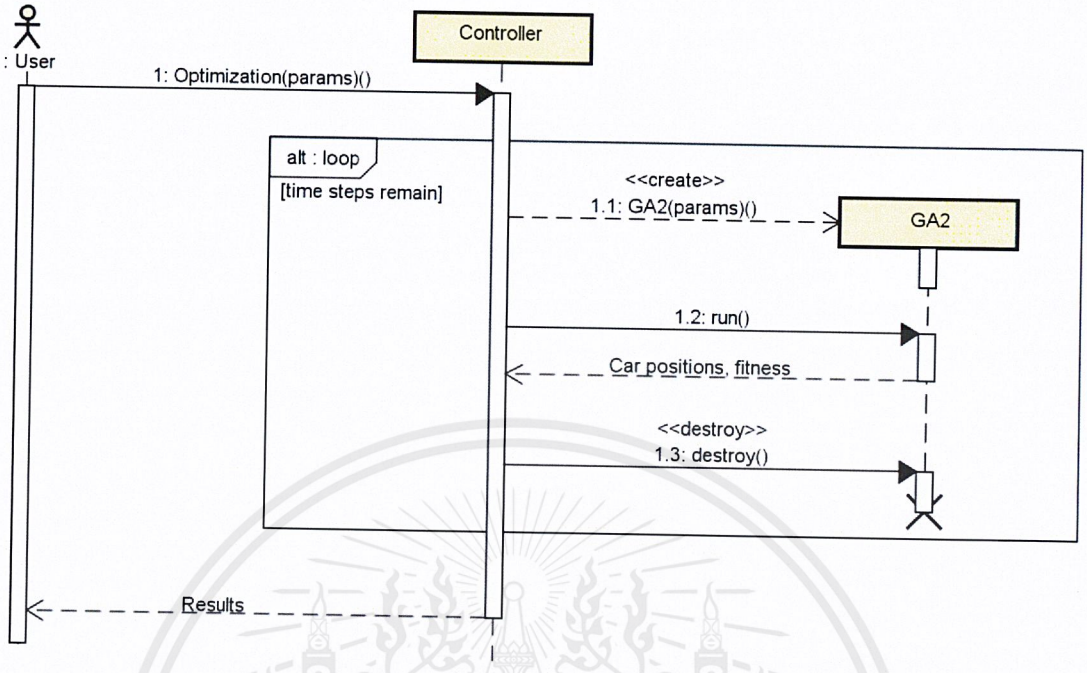


Figure 11: The sequence diagram for optimizing traffic by approach 2

The controller is one of the experiment files, or the UI application. For approach 2, we use GA2 implemented for approach 3. The implementation was made such that in the absence of the required input supplied from the output of GA1, GA2 acts as an algorithm implementing approach 2.

The sequence of actions required for optimizing traffic by approach 3 is shown as follows:

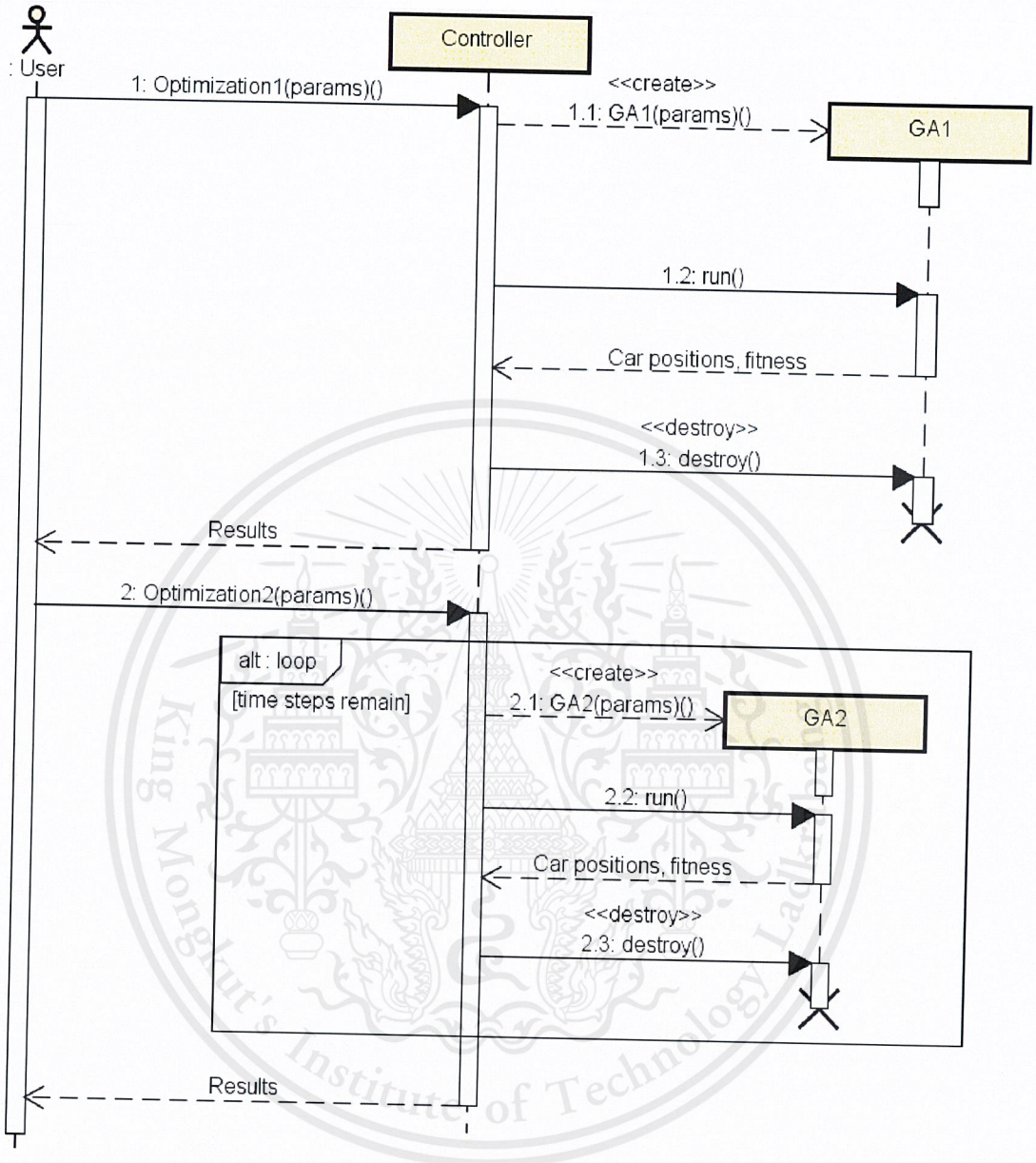


Figure 12: Sequence diagram of using genetic algorithms

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

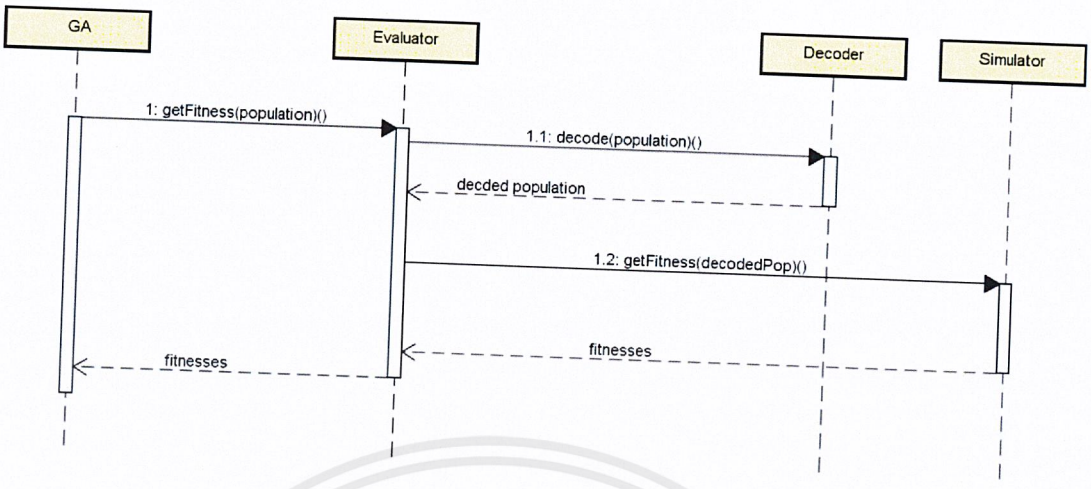


Figure 13: Sequence diagram of getting fitness values

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

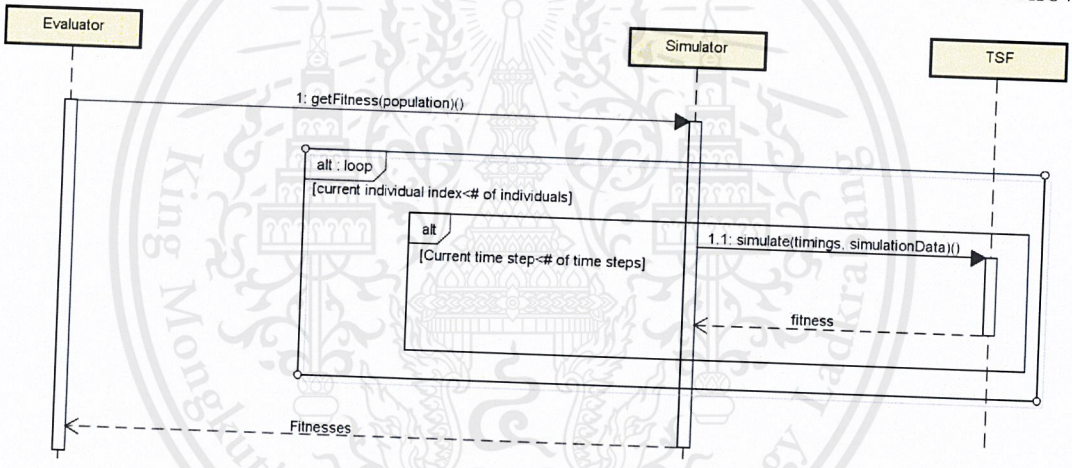


Figure 14: Sequence diagram of using TSF

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. As 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.

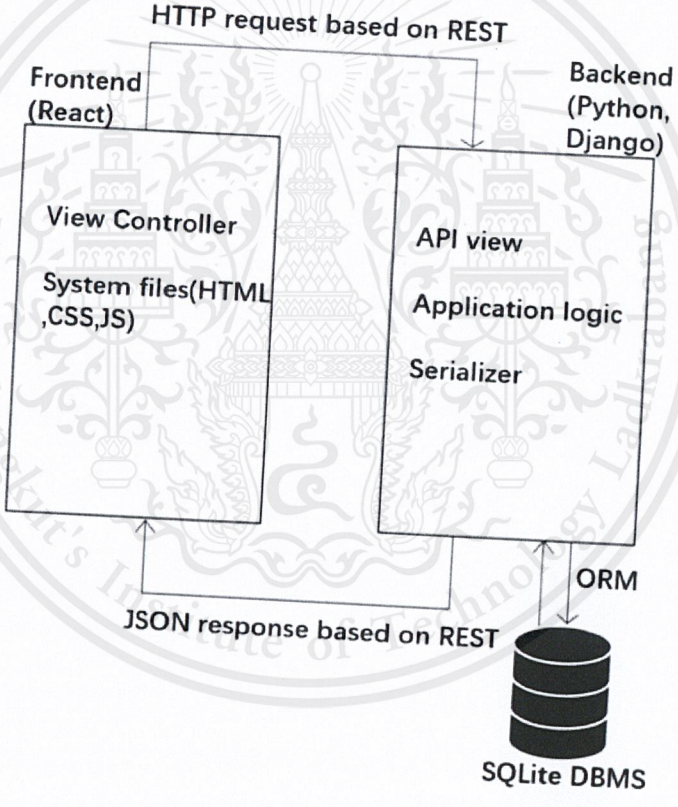


Figure 15: The application diagram

5.4.2 Use Cases

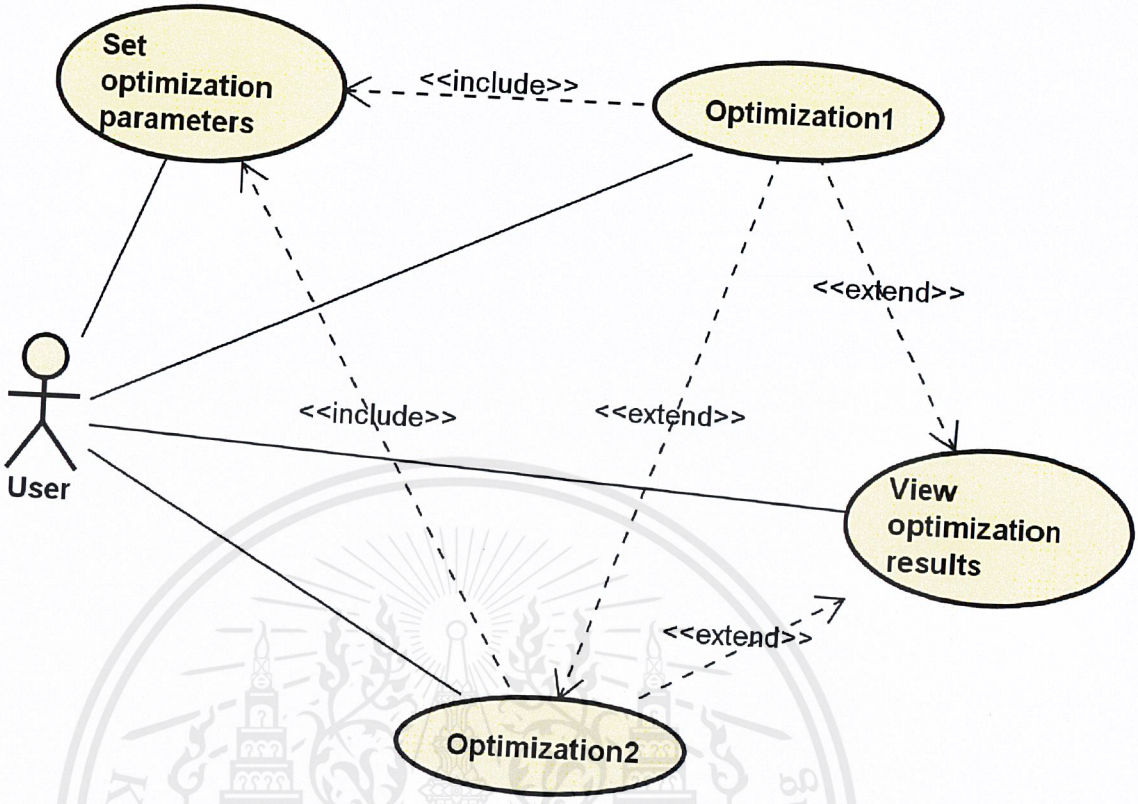


Figure 16: Use Cases

The following are the brief use cases for each process

Set optimization parameters: For each of the optimizations, the user has to supply parameters such as duration, time steps, fitness function, etc. That is done via this use case.

Optimization 1: This use case refers to optimization 1 described in chapter 4. This can be used to perform optimization via approach 1, or part 1 of approach 3. 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 view the optimization results.

Optimization 2: This use case refers to optimization 2 described in chapter 4. This can be used to perform optimization via approach 2, or part 2 of approach 3. 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 optimization results: After completing the Initialization or Real Time Optimization, the result of the optimization will be shown according to the input. The results include a map to show the state of the traffic at the end of each time step, along with graphs to show rate of convergence and numerical statistics to show the performance of the optimization.

5.4.3 Activity Diagram

A) Optimization 1 activity diagram

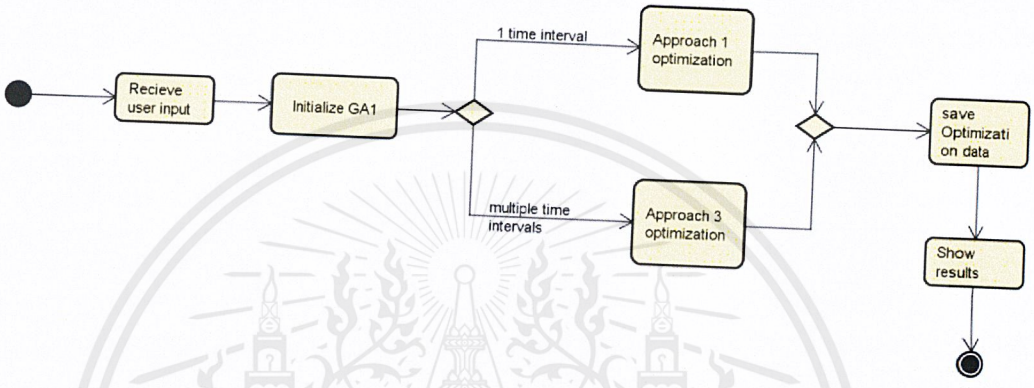


Figure 17: Initialization Activity Diagram

B) Optimization 2 activity diagram

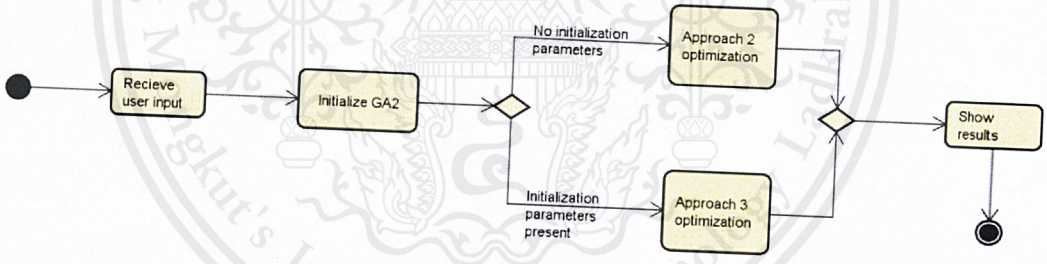


Figure 18: Real Time Optimization Activity Diagram

5.4.4 Application interface

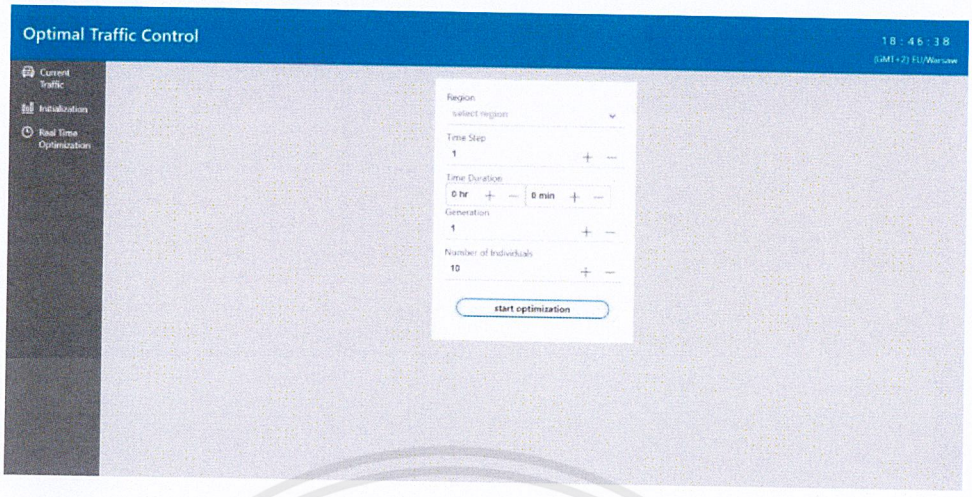


Figure 19: Optimization 1

This page shows the input page for optimization 1. The user supplies the optimization parameters such as time duration, time steps, number of GA generations etc, before beginning the simulation. At the end of this simulation, a few results (best individual) are saved in a file

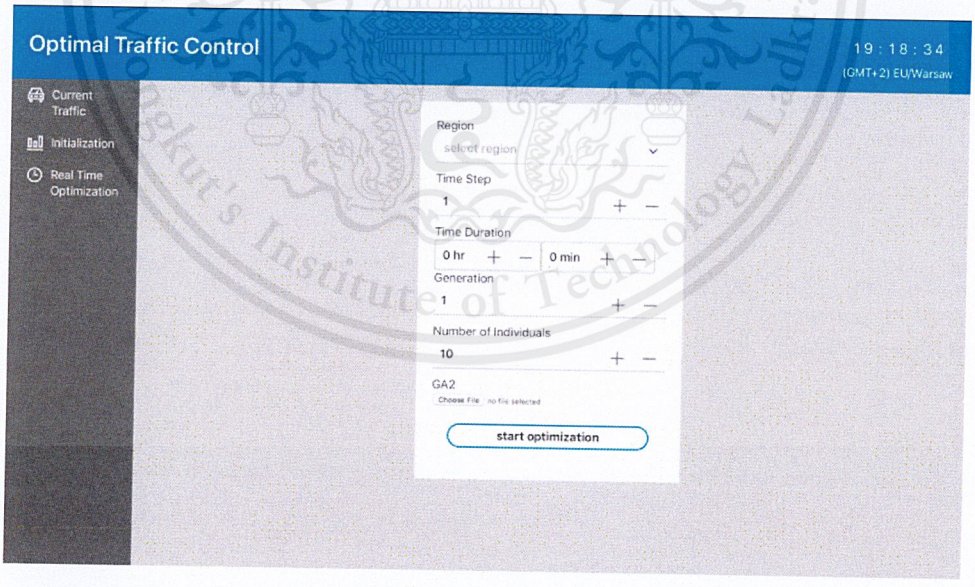


Figure 20: Optimization 2

This page shows the input page for optimization 2. Also called real time optimization, not because it is actually real time, but it could be applied for real time optimization. The user supplies optimization parameters similar to optimization 1, except for one more parameter, initialization file (file saved during optimization1). Using the initialization file makes this into approach 3, otherwise, approach 2.

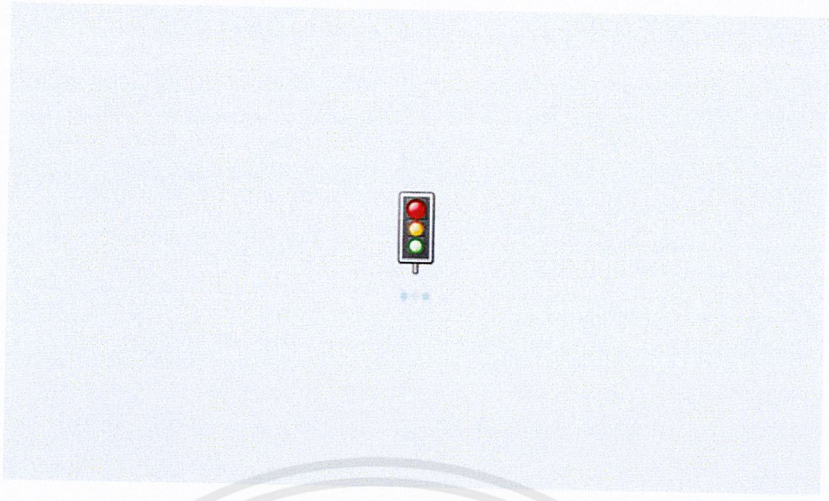


Figure 21: Loading page

The Waiting page is displayed while running the simulation. We are considering developing a better waiting page, which shows results of the optimization while the optimization is taking place.

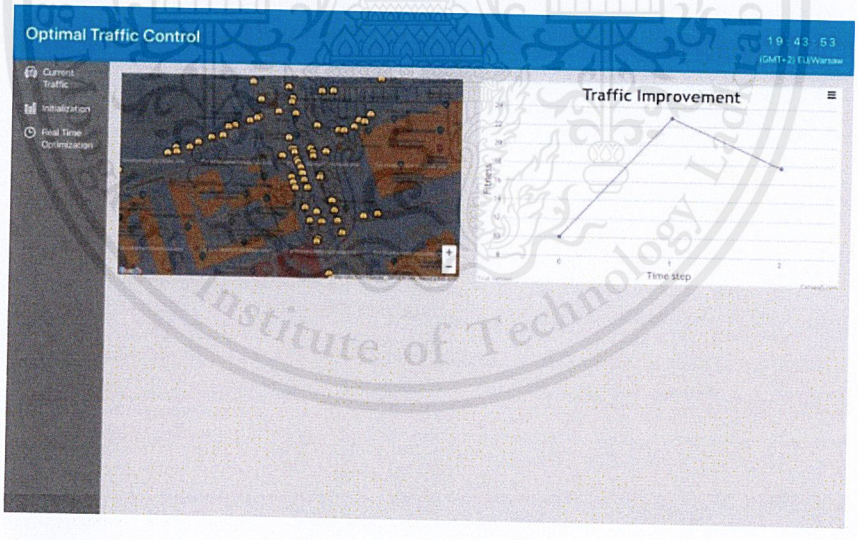


Figure 22: Result Page

The result of optimization is visualized in three ways:

- A map to show the state of the traffic at the end of each time step.
- Graphs to show the improvement in traffic conditions after time steps/generations
- Numerical statistics (not shown here)

Chapter 6

Experimental Results

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
- Acceleration: 10
- Crossroad penalty (penalty refers to the deceleration of speed): 0.25
- Turning penalty (penalty refers to the deceleration of speed): 0.5
- Traffic data: We were unable to use the traffic data mentioned in 5.2, instead, we used the default data provided by TSF

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
- Optimization duration: 6 minutes
- 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

Table 1: Grid search reuslts for GA1

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 GA2 when conducting the search are as follows:

- Generations: 3
- Individuals: 40
- Optimization duration: 2 minutes
- 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

Table 2: Grid search results for GA2

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

Although the selected operators performed best in the above presented tests, on further experimentation, it became apparent that the results were highly variable. As such the choice of mutation and crossover operators is unlikely to have a significant impact on the overall performance.

6.3 Hypotheses

As described in chapter 4, approach 3 consists of two parts. The accuracy of the first part is likely to be significantly higher; however, part 2 is necessary due to time constraints and will have a much lower accuracy. We wanted to compare the accuracy of both parts of approach 3 with the other two approached. Hence, we tested two hypotheses, comparing the two parts of approach 3 with the other approaches.

6.3.1 Hypothesis 1:

Optimization using sequences of timings (approach 3) is better than the default approaches 1 and 2. In other words, using optimization 1 alone, from approach 3 can outperform approaches 1 and 2

Experiment design:

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

- Fitness after optimization (FAO): The fitness of the system after application of the best solution obtained. Fitness is measured in total waiting time of cars.
- 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 solution, combining those worst solutions to compute improvement will greatly skew the results

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

Table 3: Experiment parameters

Note: for approach 1, the number of intervals always remains 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	Approach 1	Approach 2	Approach 3
Experiment 1	FAO: 72932 Improvement: 15.1%	FAO: 67251 Improvement: -	FAO:79809 Improvement: 11.6%
Experiment 2	FAO: 68427 Improvement:	FAO: 63182 Improvement: -	FAO: 82839 Improvement:

	19.0%		12.4%
Experiment 3	FAO: 69245 Improvement: 18.2%	FAO: 59416 Improvement: -	FAO: 80688 Improvement: 13.5%
Experiment 4	FAO: 145934 Improvement: 19.6%	FAO: 142350 Improvement: -	FAO: 172246 Improvement: 11.5%
Experiment 5	FAO: 150092 Improvement: 16.2%	FAO: 131677 Improvement: -	FAO: 168943 Improvement: 10.7%
Experiment 6	FAO: 149479 Improvement: 18.9%	FAO: 117103 Improvement: -	FAO: 163029 Improvement: 10.9%
Experiment 7	FAO: 209489 Improvement: 16.7%	FAO: 186435 Improvement: -	FAO: 258394 Improvement: 9.3%
Experiment 8	FAO: 197174 Improvement: 17.0%	FAO: 190653 Improvement: -	FAO: 248938 Improvement: 11.2%
Experiment 9	FAO: 193831 Improvement: 19.1%	FAO: 180188 Improvement: -	FAO: 241902 Improvement: 12.0%

Table 4: Experiment results for hypothesis 1

6.3.2 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 (approach 3) will be measured based on fitness after improvement, which is based on total waiting time of cars. We were unable to implement the density based fitness function described in chapter 4, as such; we will not be using that fitness function. The performance of approach 3 will then be compared with the performance of approach 1 and 2 using the same settings.

The performance of approach 3 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 20 minutes and

time steps fixed to 4, this selection was made based on the ratio of FAO to optimization duration. 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
Experiment 1	10%	68939
Experiment 2	20%	59942
Experiment 3	30%	62088
Experiment 4	40%	70418
Experiment 5	50%	62793

Table 5: Experiment results for hypothesis 2

The performance of approach 3 is very similar to that of approach 2 for the same settings (duration 20 with 4 time intervals). Approach 3 averages out to a fitness of 64836, approach 2 achieves a fitness of 67251 and approach 1 gives a fitness of 72932. More experiments should be conducted due to the high variability in the results, however, that is not possible due to the time consumed in conducting the experiments.

Chapter 7

Conclusion

We had expected approach 3 to be the best, followed by approach 1 and lastly approach 2. The reason for these expectations was that approach 3 took into account the variation of traffic flow during the considered optimization period and dealt with it by optimizing sub-genotypes to better optimize overall traffic. Approach 2 was expected to perform worst, as the optimization periods seemed too small to have a significant impact on the traffic flow.

Approach 2 proves to work the best compared against the other approaches. We believe that approach 3 was unable to perform due to the large size of its genotype. This resulted in a very large search space and the poor convergence of the genetic algorithm. Therefore, hypothesis 1 was incorrect. As far as performance of approach 1 compared to approach 2 goes, we were wrong about the duration of optimization period needed to have a significant impact on traffic flow. A much smaller interval is enough to have a significant impact on traffic. This is also demonstrated by the improving performance of approach 2 when more intervals are used to optimize the same optimization period, as shown in the experiment results.

Hypothesis 2 has neither been proven nor falsified. The performance of approach 3 is able to compete with that of approach 2, because part 2 of approach 3 is identical to approach 2. Therefore, we don't get to see whether using previously optimized traffic signals for future traffic congestions offers any significant advantage.

The fundamental idea of approach 3, which is to optimize traffic while considering its property to continuously vary can still be used to better optimize traffic; just not by genetic algorithms. Population based optimization algorithms have to spend excessive time computing fitness of individuals during optimization which has to happen in real-time. This reduces the optimization capability of population based optimization algorithms. Another approach will have to be developed based on some other optimization algorithm. Future research should be directed towards reinforcement learning or some similar optimization algorithms.

Bibliography:

[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