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

Table of Contents

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 short term adaptations to the strategy.

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

Introduction

Traffic congestion is a globally occurring problem that results in massive wastes of time. More than often, the congestion is repetitive and occurs as a result of the increase in inflow from the 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 explore two ways of combating the high computation time. Firstly, we can use a macroscopic traffic model to simulate traffic, in order to reduce computation time. Macroscopic models aggregate the description of traffic flow; macroscopic models use speed, flow and density as measures of effectiveness [2]. In the second approach, instead of having a single genetic algorithm to optimize traffic over a long period of time, we try to use a long term genetic algorithm to optimize recurring traffic congestions. The long term genetic algorithm will give us the optimal traffic density points for the roads throughout the control period. The second genetic algorithm will try to change traffic signaling to match the density points produced by the long term genetic algorithm. By this approach, we can plan over a long period of time, thereby improving rush hour traffic. Furthermore, our approach incorporates the history of traffic; hence it should be better suited at managing recurring traffic congestions.

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 10% for the area considered in [4].

1.3 Objectives

As of right now, we have established the following major objectives:

Gather travel speed data for roads in the selected region:

This data is going to be used for the long term genetic algorithm, as well as to understand the variability of traffic, which will help us improve the short term genetic algorithm.

As of right now, we have implemented and tested the scripts to collect traffic data using TomTom API. We still have to test the scripts for converting travel speed data to density data and for using density data to determine the flow of traffic at intersections.

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 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 and experiment with long and short term genetic algorithms:

In this research, we will have to experiment with the simulation parameters and different variations of genetic algorithms, to determine the best way to optimize traffic congestion. The details about the function and implementation about the genetic algorithms are given in the following section.

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.

The reviewed papers tend to 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 important 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

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].

Types of Simulation Models

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

- 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 [12].
- 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

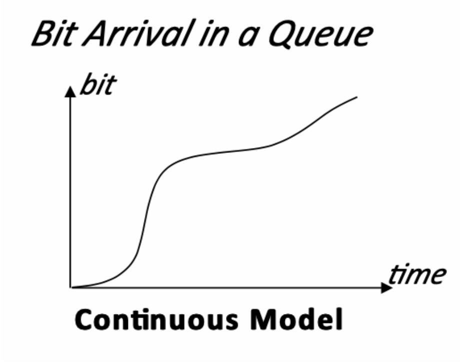


Figure 3.1 Continuous Model [6]

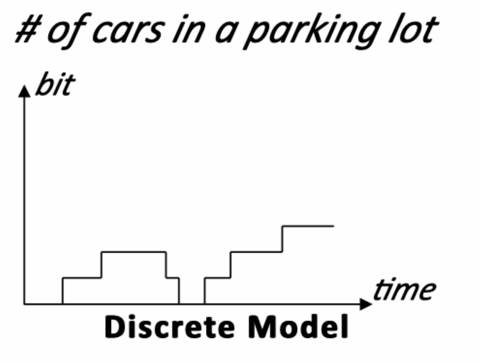


Figure 3.2 Discrete Model [6]

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 [16].

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 $\boldsymbol{v}$ is advanced by $[\boldsymbol{v} \to \boldsymbol{v}+\mathbf{1})$ if its velocity is lower than the maximum velocity and if the distance to the next car is larger than $\boldsymbol{v}+\mathbf{1}.$
- Slowing down (due to other cars): If there is a vehicle at site $\boldsymbol{i}$ and a vehicle at site $\boldsymbol{i}+\boldsymbol{j}$ and $\boldsymbol{j} \leq \boldsymbol{v}$ then the vehicle at site $\boldsymbol{i}$ reduces its speed to $\boldsymbol{j}-\mathbf{1}[\boldsymbol{v} \to \boldsymbol{j}-\mathbf{1}]$
- Randomization: The velocity of each vehicle is decreased by $[\boldsymbol{v} \to \boldsymbol{v}-\mathbf{1})$ randomly based on probability p.
- Car motion: Each car moves forward $\boldsymbol{v}$ 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 (novelty)
- Every edge is divided into identical cells (as in the NaSch model)
- Every cell may be occupied by a single car or may be empty (as in the NaSch model)
- Time is discrete (as in the NaSch model)
- Space and velocities are not discrete, cars have also positions within the cell (novelty)
- Every car has its own start and destination points chosen from a given distribution (novelty) as well as time step in which the vehicle should start its drive (novelty)
- The route of a car is calculated using a routing algorithm, for example, Dijkstra algorithm or A* algorithm (novelty)
- Every edge may consist of more than 1 lane (novelty)
- The set of roads may be divided into separate classes, roads from the same class have the same distribution of vehicles speed and the number of lanes (novelty)
- Every driver has its own profile which influences maximal car's velocity (novelty)
- Some crossroads have traffic signals, signals located at the same crossroad are synchronized (novelty)
- Drivers always tend to increase the speed (as in the NaSch model)
- Drivers reduce vehicle's velocity before the crossroad. There are parameters which determine reduction in case of turning and going straight. Cars stop before the red light (novelty).
- Vehicles may change the lane (novelty)
- Vehicles must reduce the speed if they may collide with another car (as in the NaSch model)
- Drivers may randomly reduce the speed (as in the NaSch model)
- When the vehicle's speed is calculated and established, it moves similarly to the last step of the NaSch model (with possible turn, overtaking or switching lane)

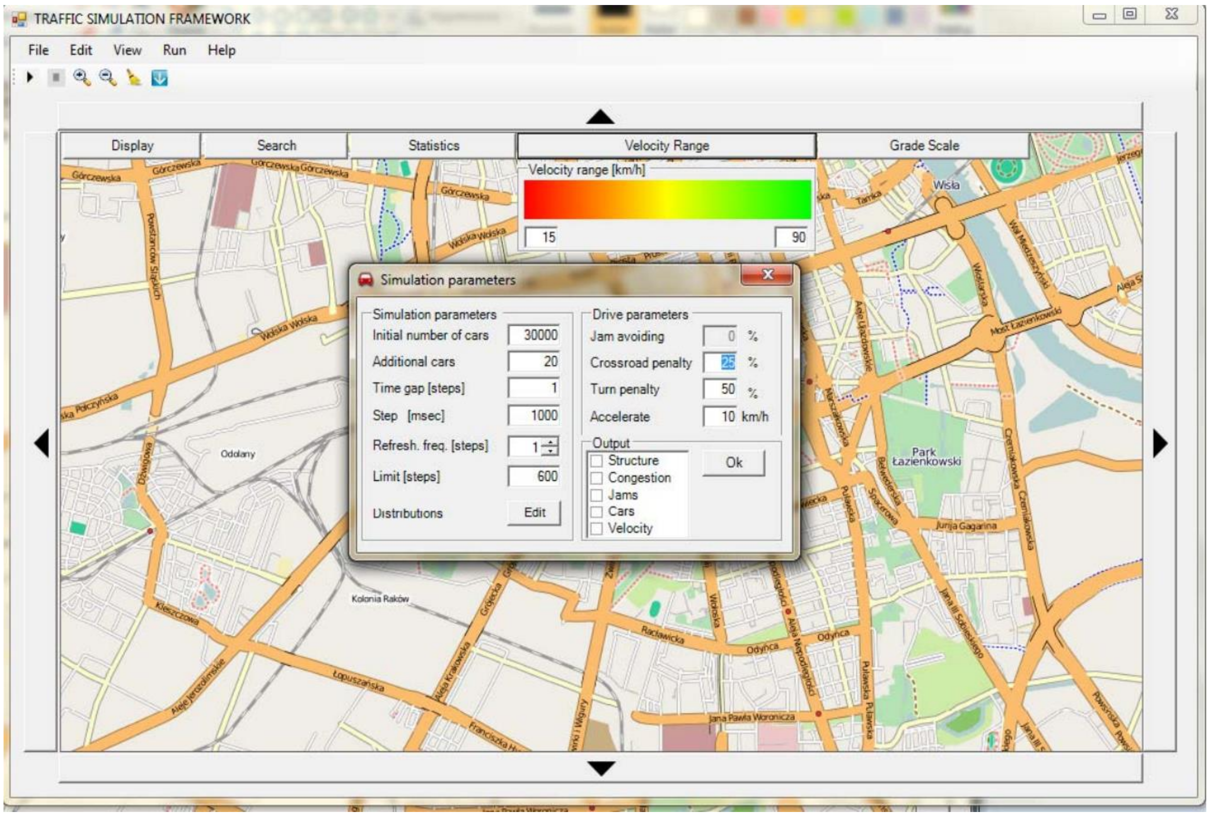
The above details are taken from [8].

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
- Simulation of vehicular traffic
- Generation of routes for drivers
- Editing settings of traffic lights
- Editing distribution of start points and destination points
- Specifying streets and areas which should be monitored during simulation
- Outputting data during simulation

The above details are taken from [9].



Simulation parameters

Initial number of cars	30000
Additional cars	20
Time gap [steps]	1
Step [msec]	1000
Refresh. freq. [steps]	1
Limit [steps]	600
Distributions	Edit

Drive parameters

Jam avoiding	0	%
Crossroad penalty	25	%
Turn penalty	50	%
Accelerate	10	km/h

Output
- ☐ Structure
- ☐ Congestion
- ☐ Jams
- ☐ Cars
- ☐ Velocity

Figure 1: *Editing simulation parameters and starting simulation, taken from [8]*

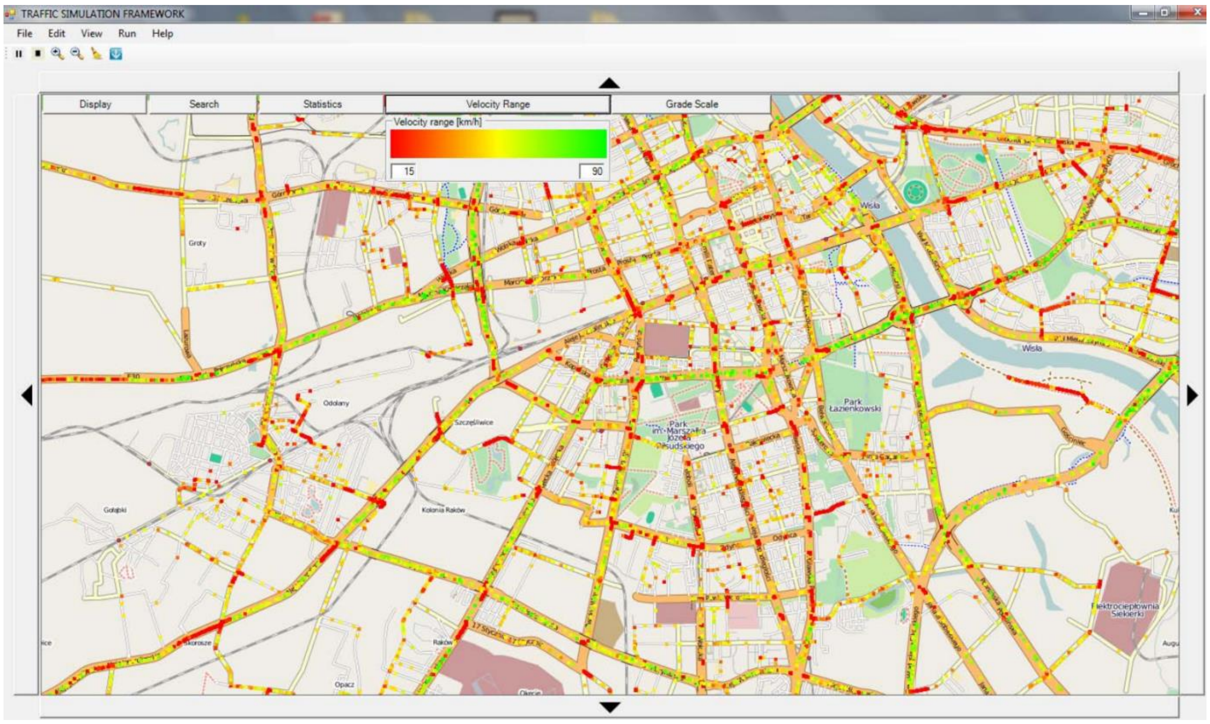


Figure 2: *The main window of the TSF software, taken from [8]*

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).

- 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 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 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 to try to mimic the density points outlined by the long term genetic algorithm, for traffic conditions similar to the one optimized by the long term genetic algorithm.

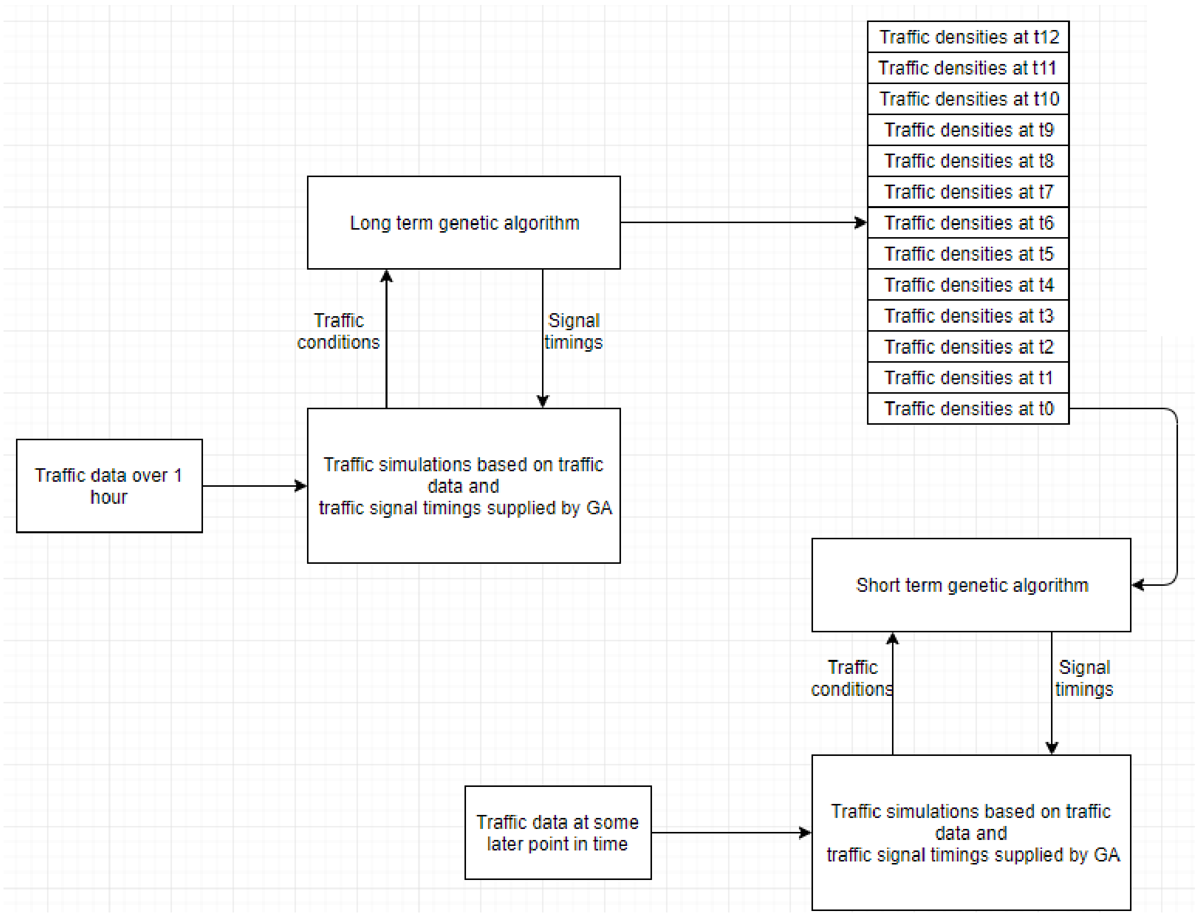


Figure …: Diagram showing the operation of proposed method

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:

- Setup a model that can accurately predict traffic based on the changing inflow and outflow of traffic from each road.
- Select a region to model the traffic
- 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 Long term genetic algorithm (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 the traffic signals over the entire period. The output of this process is going to be the optimal traffic densities at each road after small time intervals. This task is computationally expensive and will have to be done prior to the real time control task.

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 the long term genetic algorithm.

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 $r(A)\colon \forall A_i \in A_r(A_i) = r(A)$. 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 $N = \{(min_g, min_r), (min_g + 1, min_r), (min_g, min_r + 1), \dots, (max_g, max_r)\}$ be the set of **possible phase durations** for the red and green traffic signals, where max_g, max_r, min_g and min_r correspond to the maximum and minimum phase durations of red and green signals respectively.*

Definition 4: *Let $T = \{t_1, t_2, t_3, \dots, t_{num\_steps}\}$ 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 $GN = \{g_1, g_2, g, \dots, g_{num\_steps}\}$ 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 $Genotype\colon GN \to T$.*

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:

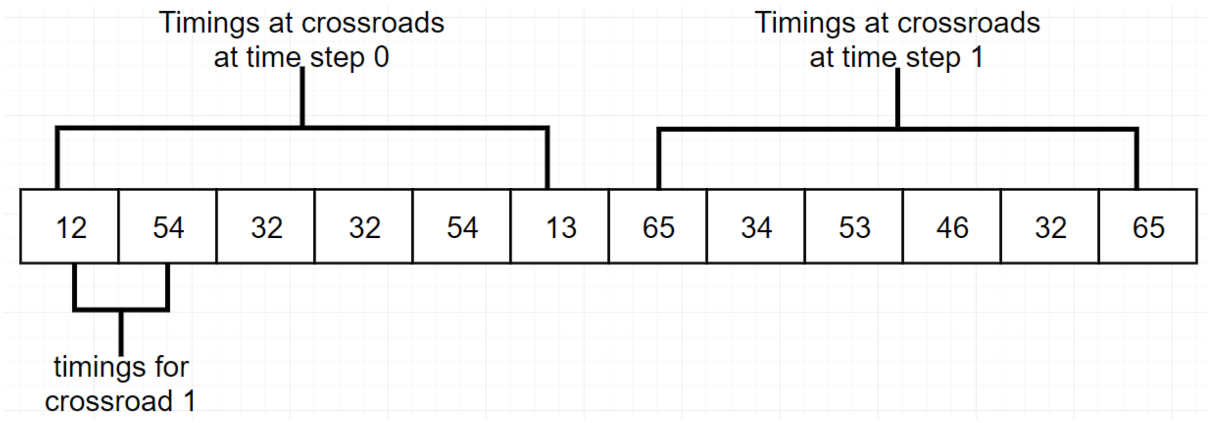

Timings at crossroads at time step 0
Timings at crossroads at time step 1
12
54
32
32
54
13
65
34
53
46
32
65
timings for crossroad 1

Figure 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:

If there are N individuals in the population, we will take $\sqrt{N}$ of the individuals with the best fitness score.

Crossover:

Each selected individual will be crossed with every other selected individual. When two individuals are crossed, the new individual will have genes randomly selected from either of the parents, with equal probability.

Mutation:

Each gene in the newly created individuals will be randomly changed to some random value from the set of possible values for signal timings using probabilities from the set {0.001, 0.01, 0.1} in successive experiments (different probability in different experiments).

Termination:

We will not terminate the genetic algorithm based on the fitness of the individuals, but based on the computation time; we are yet to determine the time limit.

Simulation Specifications:

The specifics for the simulation that the long term genetic algorithm will try to optimize are as follows:

- Time duration: 60 minutes (we will consider longer durations later)
- Time step duration: 5 minutes
- Time steps (n): (Time duration)/(Time step duration) = 12

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 the best genotype obtained from the long term genetic algorithm. Let DTT be the set of densities at each road in the network at each time step* $DTT = \{D^1, D^2, D^3, \dots, D^{num\_steps}\}$*. Then the* ***output*** *of the algorithm is DTT.*

4.3 Short term genetic algorithm (GA2):

The short term genetic algorithm will be used to achieve real time control of traffic. Instead of trying to find the traffic signal timings to get optimal traffic densities like in GA1, this genetic algorithm focuses on finding traffic signal timings that will reproduce the traffic densities produced by the long term genetic algorithm. Hence the performance measure for this genetic algorithm will be the difference between the simulated traffic densities produced by the individuals in the population and DTT.

Solution domain:

Definition 6: *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* $N = \{(min_g, min_r), (min_g + 1, min_r), (min_g, min_r + 1), \dots, (max_g, max_r)\}$ *be the set of possible phase durations for the traffic signals, where* max_g, max_r, min_g *and* min_r *correspond to the maximum and minimum durations of red and green phases respectively.* ***Genotype*** *for the road network is any function* $Genotype: G \to N$ *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:

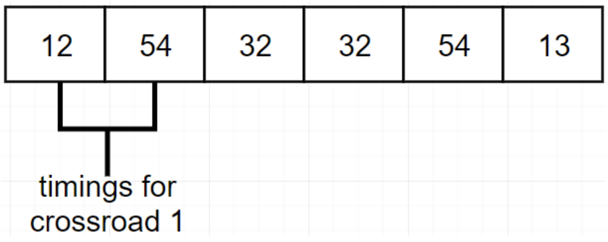


Figure 4: Genotype with 3 crossroads in GA2

Fitness function:

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,* $D = \{D_1, D_2, D_3, \ldots, D_n\}$*, 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 the fitness functions defined in section 4.2

Selection:

If there are N individuals, we will select $\sqrt{N}$ of the individuals with highest fitness score.

Crossover:

Each selected individual will be crossed with every other selected individual. When two individuals are crossed, the new individual will have genes randomly selected from either of the parents, with equal probability.

Mutation:

Each gene in the newly created individuals will be randomly changed to some random value from the set of possible values for signal timings using probabilities from the set {0.001, 0.01, 0.1} in successive experiments (different probability in different experiments).

Termination:

We will not terminate the genetic algorithm based on the fitness of the individuals, but based on the computation time; we are yet to determine the time limit.

Output:

The output for the short term genetic algorithm is going to be the traffic signal timings to make the traffic as similar as possible to the traffic simulated in the long term genetic algorithm.

4.4 Experiments

In order to establish a baseline, we need to perform the following experiments:

Baseline Experiment1: Use GA1 to optimize traffic over an interval of 5 minutes, i.e. use GA1 to optimize traffic over a time duration of 5 minutes with 1 time step. Similar to the method proposed in [3]. This experiment is meant to serve as a baseline for the performance of GA2, when using fitness function 3.

Baseline Experiment2: Use GA1 to optimize traffic repeatedly over small 5 minute intervals for a period of an hour (12 simulations of 5 minutes with different traffic signal settings) instead of using GA1 to optimize traffic for an hour in just one iteration (simulations of an hour that tests different combinations of traffic signal settings). This experiment is meant to serve as a baseline for the performance of GA1 and our proposed method

In our research, we are going to experiment with the following factors and parameters to improve traffic conditions:

- Variability in traffic conditions between simulations used by GA1 and GA2
- Time step duration
- Number of time steps considered by GA1
- Fitness functions for GA1 and GA2
- Genetic algorithm parameters
- Area whose traffic signals are optimized by genetic algorithms
- Area used to calculate fitness: If this area is smaller than the area used to optimize traffic signals, the fitness of the solutions should be higher. We expect this because of the results given by [4]
- Effect of optimization of a fitness function on other fitness functions

We define the following specifications for the parameters to be experimented with:

Fitness function1: Time spent travelling under 20km/h, can be used by both GA1 and GA2

Fitness function2: Time spent not moving, can be used by both GA1 and GA2

Fitness function3: Difference between densities at roads between solutions obtained by GA1 and GA2 at the same time step. Used only by GA2, specifics in section 4.3

Fitness function4: weighted fitness (the fitness value of a certain region computed by using fitness functions 1, 2, or 3 is multiplied by a certain weight)

Area A:

Area B:

There are far too many experiments to be listed; hence we will only mention experiments that yielded interesting results in section 6.

Chapter 5: Development/Implementation

In this section, we describe the experiments conducted and 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. We haven't however fixed the values for the look up table yet
- 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. 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 simulation

[Left for next semester]

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:

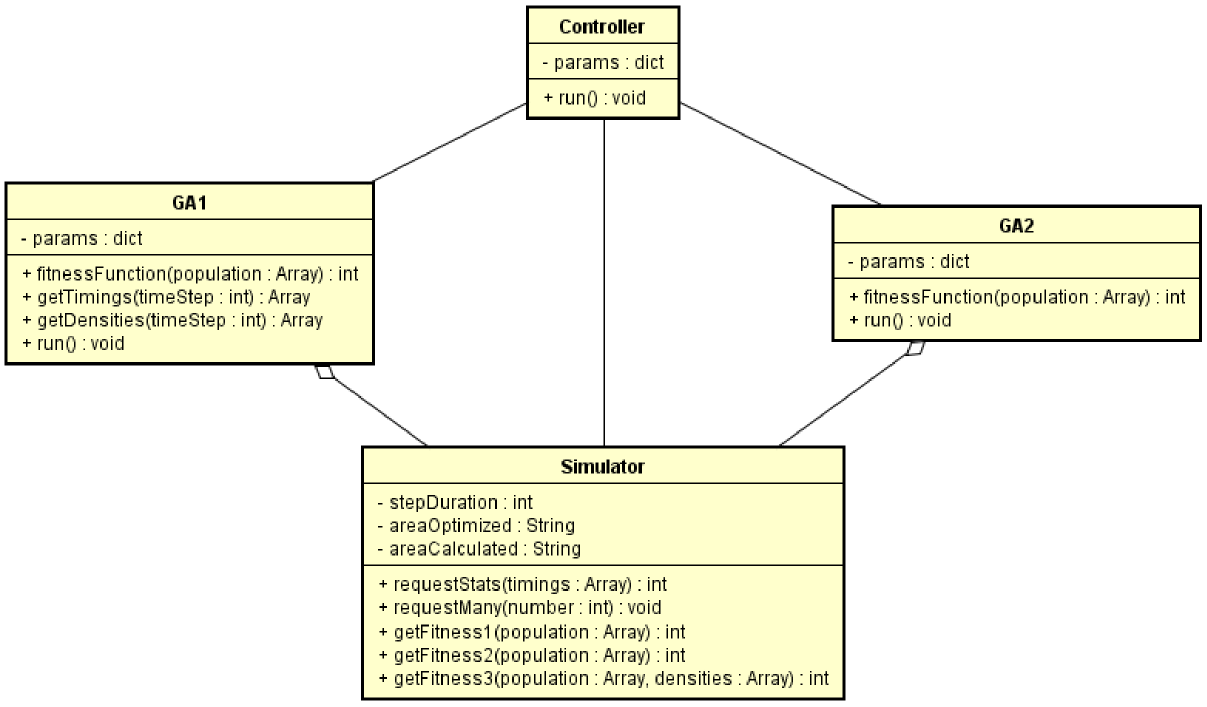
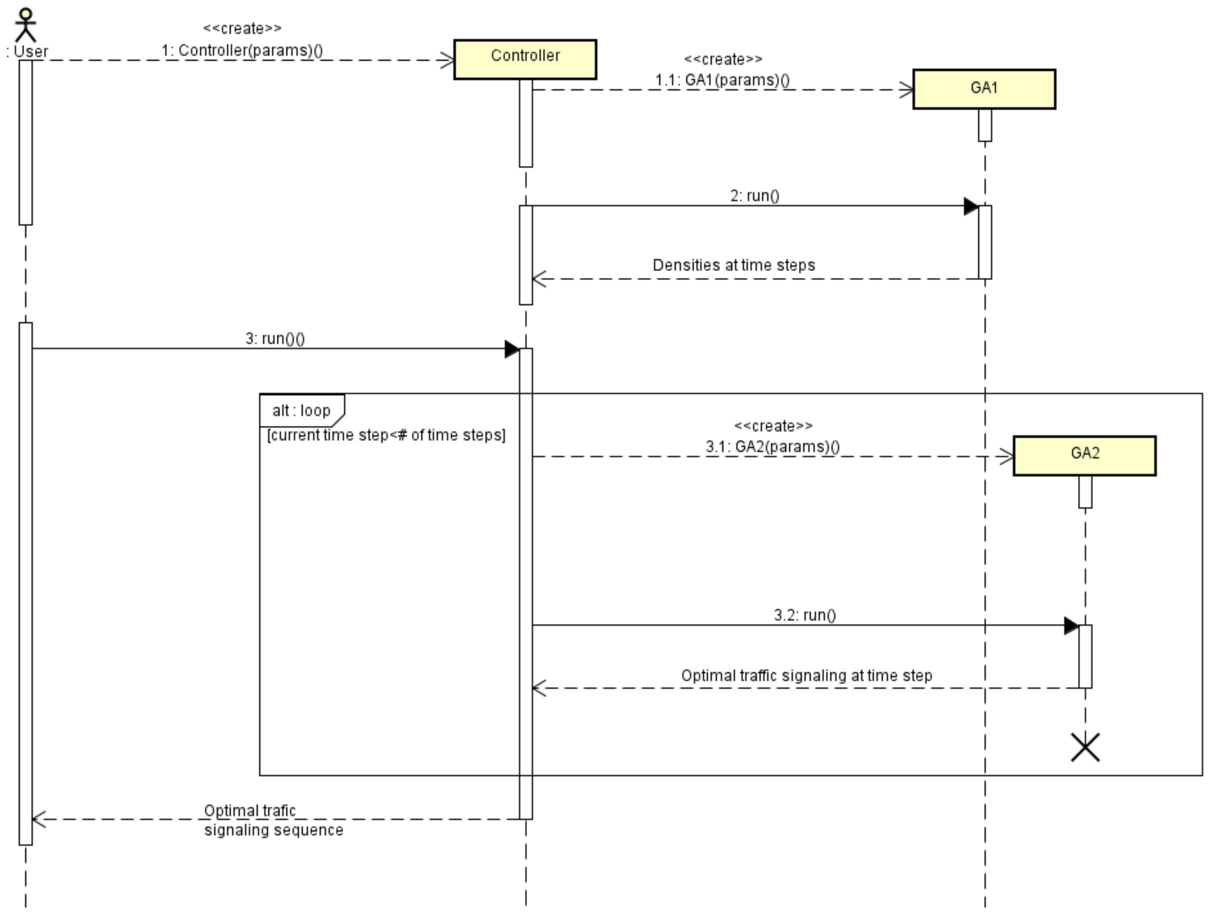

Controller
- params : dict
+ run() : void
GA1
- params : dict
+ fitnessFunction(population : Array) : int
+ getTimings(timeStep : int) : Array
+ getDensities(timeStep : int) : Array
+ run() : void
GA2
- params : dict
+ fitnessFunction(population : Array) : int
+ run() : void
Simulator
- stepDuration : int
- areaOptimized : String
- areaCalculated : String
+ requestStats(timings : Array) : int
+ requestMany(number : int) : void
+ getFitness1(population : Array) : int
+ getFitness2(population : Array) : int
+ getFitness3(population : Array, densities : Array) : int

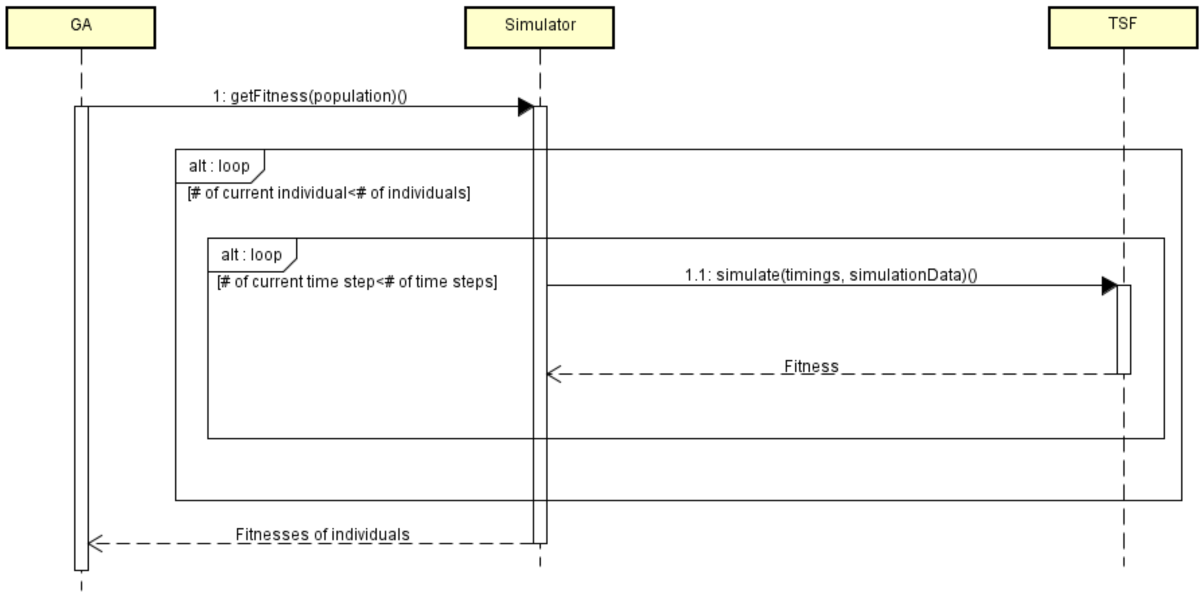
Figure ...: class diagram of genetic algorithms

The typical sequence of actions is shown as follows, the actions required to setup are excluded as it has not been implemented (we have only been experimenting with the default traffic conditions provided by TSF):



Figure…: Sequence diagram of using genetic algorithms

The sequence of actions by which the genetic algorithms determine the optimal settings is as follows:



GA
Simulator
TSF
1: getFitness(population)()
alt : loop
[# of current individual<# of individuals]
alt : loop
[# of current time step<# of time steps]
1.1: simulate(timings, simulationData)()
Fitness
Fitnesses of individuals

Figure…: Sequence diagram of running simulation

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 could go up to a month; 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, 32 jobs can be executed in parallel.

5.4 GUI application

[Left for next semester]

Chapter 5: Experimental Result

Baseline Experiment 1:

Evolution step	Fitness 1	Fitness 2	Fitness 4

Overall improvement from first to last evolution step:

Improvement in last step compared to traditional traffic signal setting:

Baseline Experiment 2:

Evolution step	Fitness 1	Fitness 2	Fitness 4

Improvement in fitness for best solution between last and first time step:

Overall improvement from first to last evolution step:

Improvement in last step compared to traditional traffic signal setting:

Chapter 7: Current Progress & Project Plan for Next Term

Current progress:

- Implemented scripts to gather traffic data
- Implemented genetic algorithms
- Parallelized requests to reduce time consumption

Plan for next semester:

- Alter the simulation based on traffic data, both for initialization and real time modifications. Currently, we can only obtain information about relative flow at intersections, whereas TSF requires information about starting point and destination of each car. For this task, we are going to have to randomly generate routes for cars based on the probabilities derived from the data we gathered about traffic flow. A challenge for this task is going to be to supply these paths to TSF as it is not an open source program, and we do not have direct access to its code
- Identify the degree by which traffic conditions vary on a daily/weekly basis to replicate this variability in the simulations. Our approach is based on the assumption that some traffic congestions are repetitive in nature. In order to make sure that our research is valid, we need to find traffic congestions in real life that have a tendency to repeat and measure the degree by which they vary. These variations will have to be replicated in TSF to fully test our approach.
- Implement fitness functions 3 and 4. Fitness function 3 is a major part of the novelty of our research. However, we can't yet implement it because it requires data about the entire traffic state of the simulation, and currently, TSF does not supply such data. We will have to request certain changes to TSF before we can implement fitness function 3. Fitness function 4 also requires the same data and can't be used without data about the traffic state.
- Implement a GUI to display alterations to traffic: Lastly, we are going to have to implement a GUI that can display the improvements in traffic congestion through the different generations of the genetic algorithm. For this task, we are also going to need the traffic state data from TSF.

References:

[1] N. T. Ratrout and S. M. Rahman, "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, 2009.

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

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

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

[5] Kwasnicka, H., Stanek, M.: Genetic Approach to Optimize Traffic Flow by Timing Plan Manipulation. In: Yuehui, C., Abraham, A. (eds.) *ISDA 2006 Proceedings*, vol. II, pp. 1171–1176. IEEE Computer Society, Los Alamitos (2006)

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

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

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

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

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