Miguel Ángel BARRERO DÍAZ

Adaptive traffic control system based on genetic algorithms

FINAL MASTER'S PROJECT

Directed by Dr. Jordi DUCH GAVALDÁ

University Master's Degree in Computer Security Engineering and Artificial Intelligence

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

September 3, 2021

"There is a driving force more powerful than steam, electricity and nuclear power: the will."

Albert Einstein

UNIVERSITAT ROVIRA I VIRGILI

Abstract

ETSE-School of Engineering
DEIM

University Master's Degree in Computer Security Engineering and Artificial Intelligence

Adaptive traffic control system based on genetic algorithms

by Miguel Ángel BARRERO DÍAZ

From the beginning of the road traffic, the infrastructures and regulations have evolved with the objective of being adapted to the increasingly volume of vehicles and its interaction with pedestrians. Traffic lights were developed aimed to automate the traffic regulation in road intersections and the police officers were replaced by traffic lights geared by automated regulators. This solutions have evolved towards complex systems where many parameters are taking into account and are ,in general, managed from a control center. Usually, these kind of infrastructures has large implementation costs and maintenance expenses and are not feasible in many cases. This paper proposes a solution based on genetic algorithms were each traffic light regulator has it own intelligent agent, which learns from traffic data and takes time distribution decisions in a decentralized way, based only on its own parameters taken from the environment.

Acknowledgements

I will be eternally grateful to all my family, especially, my wife María, for her patient along these years of sacrifice, many thanks to my master's thesis supervisor Jordi Duch, for his support in order to achieve the objectives of this work.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
DNN	**D**eep **N**eural **N**etwork
GA	**G**enetic **A**lgorithm
GNA	**G**enerative **A**dversarial **N**etwork
HOF	**H**all **O**f **F**ame
ITS	**I**ntelligent **T**ransport **S**ystem
LoS	**L**elvel **O**f **S**ervice
MA50	**M**oving **A**verage **50**
NN	**N**eural **N**etwork
NSGA-II	**N**on **D**ominated **S**orting **G**enetic **A**lgorithm
TL	**T**raffic **L**ights
UK	**U**nited **K**ingdom
UTC	**U**rban **T**raffic **C**ontrol
V2V	**V**ehicle **T**o **V**ehicle
V2I	**V**ehicle **T**o **I**nfraestructure

Physical Constants

Vehicle Length	$V_l = 4.27\,\mathrm{m}$
Safety distance	$S_d = 4.57\,\mathrm{m}$

List of Symbols

DRe	Delay rate	$\mathrm{s\,m^{-1}}$
DRo	Delay ratio	no units
L_s	Segment length	m
L_v	Average vehicle length occupancy	m
N_l	Lanes quantity	Number of lanes
N_v	Spatial mean volume	vehicles
N_{max}	Max number of vehicles allocatable	vehicles
R_i	Degree of road segment congestion	
R_{NC}	Proportion of no congested state	no units
SPI	Speed Performance Index	no units
SRI	Speed Reduction Index	no units
T_{ac}	Actual travel time	s
T_{ff}	Free flow travel time	s
t_{NC}	Duration of no congested state	s
t_t	Period of observation	s
TR_{ac}	Actual travel rate	$\mathrm{s\,m^{-1}}$
TR_{ap}	Acceptable travel rate	$\mathrm{s\,m^{-1}}$
TR	Travel rate	$\mathrm{s\,m^{-1}}$
T_t	Travel time	s
v_{ac}	Actual travel speed	$\mathrm{m\,s^{-1}}$
v_{avg}	Average travel speed	$\mathrm{m\,s^{-1}}$
v_{max}	Maximum permissible road speed	$\mathrm{m\,s^{-1}}$
v_{ff}	Free flow speed	$\mathrm{m\,s^{-1}}$
VC_r	Volume to capacity ratio	no units

To my land, Galicia, where nevertheless, the sun shines everyday.

Chapter 1

Introduction

1.1 Historical notes on traffic lights

The importance of traffic management has increased proportionally to the number of vehicles flowing on the roads, the importance of deliver goods in time, the coexistence of vehicles and pedestrians between many other factors has made the UTC a very broad study field.

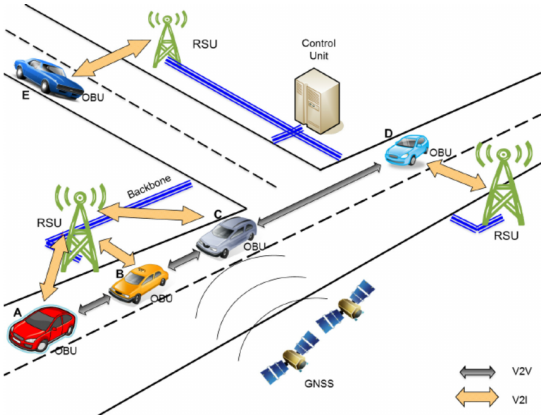
An important tool to manage the traffic along this years were the TL (Traffic Lights). We are going to take a glance to the different phases during the evolution of this kind of systems[1] :

- The first system of this class was deployed in 1868 and wasn't automatic, it needed to be manipulated by the police officer and was in operation only for a month.
- Next, the former electric and automatic TL system were installed in the UK during 1920s after the success of its implementation in other countries as United States or Germany. During this period that last until the 1980s, the phase time duration was fixed, this design limitation makes the TL not suitable for the increasingly traffic demand.
- The next period to consider, overlaps the previous one, since may be considered to begin in 1970s and last to the present days. This type of system is defined by the use of vehicle actuated isolated junctions. This technology was geared by the use of induction loops, which are able to detect the presence of vehicles, triggering in this way the green time per phase can be assigned in function of the traffic demand.
- The same time that the previous milestone was achieved, the vehicle first actuated coordinated junctions were implemented. This systems, as the seen before, run under the same operating concept, the difference stands in the fact, that in this case, exists a coordination between the different TL controlled junctions. This architectures are broadly used yet, although, this is changing quickly.
- Around 1997 ushered the primitive ITS (Intelligent Transport Systems) solutions merged with available UTC (Urban Traffic Control) architectures at that moment. This infrastructures brought many advantages, as more safety, the integration of public transport in the system, environmentally sustainability, between others. The standardisation efforts of ITS by the authorities is bringing a higher possibility of integration of new solutions, the UTMC (Urban Traffic

Management and Control) or the NTCIP (National Transportation Communications for ITS Protocol) from UK and United States, respectively, are examples of this normalization attempts.

- Nowadays,there are arising promising technologies powered by the mobile networks, the Internet of things or the AI, just to mention some of it. This solutions can combine several infrastructures [2] ,bringing to us novel solutions,as V2V (Vehicle to Vehicle) or V2I (Vehicle to Infraestructure), more sustainable and efficient.

FIGURE 1.1: Modern V2V/V2I Infrastructure [2]



1.2 Traffic congestion

The traffic congestion is a direct consequence of the population rise, the greater amount of vehicles and the proliferation of delivery services [3]. The congestion state is reached when the number of vehicles surpasses the road capacity, interrupting in this way, the traffic flow. The traffic congestion has a great impact in populated areas, causing an increase in the delay travel times and, as a consequence, high costs and a severe environmental impact. Fighting against this problem requires great efforts by the public and private actors. It is known that the first step in building an effective system is the observation of the traffic congestion. The measurement of traffic conditions is a key factor to carry out prevention measures in order to avoid congestion [3].

1.2.1 Causes of congestion

Multiple reasons are begin the congestions, incidents, weather-related events, works in the road, signals ...
this leads to a differentiation between two types of congestion [3].

Recurring congestion

This happens primarily in urban areas or metropolitan areas at the peak hours, this type of recurrent events causes the half of the hours that users spend in congestions [3]. The main causes of recurrent congestion 1.1 are, the bottlenecks, when the demand exceeds the road capacity, insufficient infrastructure with respect to the population in a certain area 1.2, the variability in the traffic flows, which can change

TABLE 1.1: Congestion measurement indices [3]

Velocity	Travel Time	Delay	Level of Service	Congestion Indices
Speed Reduction Index Speed Performance Index	Travel Rate	Delay Ratio Delay Rate	Volume to Capacity Ratio	Relative Congestion Index

from day to day and wrong designed infrastructure, with bad time distribution in the TL junctions, bad chosen stop signs etc... ...

Non recurring congestion

This kind of congestion are caused by isolated events, as traffic accidents, that can block some of several lanes causing the reduction of the capacity, weather conditions, that affects the road conditions and the drivers behaviour, sometimes, due to special events, as for example a football match, the traffic conditions differs from usual, this occasional peaks could cause an overwhelming vehicle flow that the infrastructure is not able to manage.

1.3 Measuring congestion

As I have mentioned earlier, measuring the congestion is a key point in fighting against it. The congestion measures are based in several parameters, they are the velocity, the travel time, the delay, the level of service and the congestion indices [3].

1.3.1 Speed Reduction Index

This index is the ratio between of the relative speed change between congested and free flow conditions as the defined in the equation below [3]:

$$SRI = \left(1 - \frac{v_{ac}}{v_{ff}}\right) \times 10 \tag{1.1}$$

The v_{ac} and v_{ff} variables are measuring, respectively, the actual travel speed and the free flow speed.

1.3.2 Speed Performance Index

This value goes from 0 to 100, using this parameter, the traffic can be viewed as a linguistic variable,the set of instances classifies the traffic state level in four categories threshold at 25,50 and 75 percent [3]. The parameter is obtained using the ratio between the average travel speed (v_{avg}) and the maximum permissible road speed (v_{max}).

$$SPI = \left(\frac{v_{avg}}{v_{max}}\right) \times 100 \tag{1.2}$$

TABLE 1.2: Speed performance index intervals [3]

Speed Performance Index	Traffic State Level
(0,25)	Heavy congestion
(25,50)	Mild congestion
(50,75)	Smooth Higher
(75,100)	Very smooth

1.3.3 Travel Rate

This parameter is the rate of motion, in other words, the necessary time to travel and unit of distance.

$$TR = \frac{T_t}{L_s} \tag{1.3}$$

Being T_t the travel time and L_s the segment length.

1.3.4 Delay Rate

With this parameter we can obtain the difference in time for a vehicle to travel a certain road segment between the actual condition and the ideal one.

$$DRe = TR_{ac} - TR_{ap} \tag{1.4}$$

1.3.5 Delay Ratio

The delay ratio is useful to compare the relative congestion levels in different roads, and It is obtained with the equation below.

$$DRo = \frac{DRe}{TR_{ac}} \tag{1.5}$$

1.3.6 Volume to Capacity Ratio

In order to measure the LoS (Level of Service), this parameter may be used. The LoS classes can be obtained in function of this ratio,splitting it in several categories.

$$VC_r = \frac{N_v}{N_{max}} \tag{1.6}$$

Where the spatial mean volume N_v is obtained as:

$$N_v = \frac{V_{count}}{t_t} \tag{1.7}$$

Being V_{count} the number of vehicles and t_t the period of observation.

Where the max number of vehicles allocatable N_{max} is:

$$N_{max} = \frac{L_s}{L_v} \times N_l \tag{1.8}$$

Being L_s and L_v ,respectively, the segment length and the average length vehicle occupancy and N_l the number of lanes in the road.

TABLE 1.3: Level of Service in traffic environments [3]

LoS Class	VC_r **parameter value range**
A→ Free flow	(0,0.6)
B→ Stable flow with unaffected speed	(0.61,0.7)
C→ Stable flow but speed is affected	(0.71,0.8)
D→ High-density but the stable flow	(0.81,0.9)
E→ Traffic volume near or at capacity level	(0.91,1)
F→ Breakdown flow	>1

And where L_v is obtained using:

$$L_v = V_l + S_d \tag{1.9}$$

being V_l the vehicle length and S_d the security distance, which parameters are stated as constant values 1.

In general, as we state in the previous equation, the vehicle length and the spatial separation are taking into account to obtain the average lane vehicle occupancy. The LoS values and ranges can be seen in table 1.3

1.3.7 Relative Congestion Index

A very low value of this index implies the absence of congestion ,conversely, values greater than two are pointing a great congestion level,It is defined as:

$$RCI = \left(\frac{T_{ac} - T_{ff}}{T_{ff}} \right) \tag{1.10}$$

Where T_{ac} is the actual travel time and T_{ff} is the free flow travel time, which can be obtained as:

$$T_{ff} = \frac{L_s}{v_{ff}} \tag{1.11}$$

Being v_{ff} the free flow speed

1.3.8 Road Segment Congestion Index

This measure is obtained taking as a reference the normal traffic conditions, the no-congestion state is those where the speed performance index(SPI) is higher than 50. The R_i index takes values between 0 and 1, values closed to 0 are related with higher congestion levels, it is obtained using the average speed performance index SPI_{avg} and the proportion of the non congested state R_{NC}.

$$R_i = \frac{SPI_{avg}}{100} \times R_{NC} \tag{1.12}$$

Where:

$$R_{NC} = \frac{t_{NC}}{t_t} \tag{1.13}$$

Being t_{NC} the duration of no congested state.

1.4 TL basic concepts

At this point I will explain some basic concepts related with TL junctions which are very important to lay the foundations for the upcoming sections.

- **Group:**A group is set of traffic lights which share the colors of its lights every time.
- **Stable stage:**Also known as stable phase, is a stable combination of lights in every groups of a TL junction. The stable stage determines the possible movements of vehicles and pedestrian into the junction, the time duration may be modified to achieve a better TL performance.
- **Transitory stage:**It corresponds with the light states when the TL is passing from a stable stage to another, the transitory times are fixed by design and we can't change it.
- **Structure**: The structure defines the sequence in what the different stages are placed.
- **Cycle:**It corresponds with the time that takes to complete full sequence, i.e. the sum of all stable and transitory stage times
- **Position:**The position is each of the light combination in the groups in a transitory stage. This duration,as we stated earlier is fixed.
- **Distribution:**This concept is related with the available time of the cycle assigned to every stable phases.
- **Offset:**It is the delay of the begin of the cycle with respect to a reference.
- **Plan:**A plan is defined by a combination of a structure,a cycle and a distribution. This parameters are are kept while the TL is executing one plan.

1.5 Decentralization

Traditionally, centralized approaches were used to solve the congestion problems and traffic management. Novel solutions are trying to use a decentralized model where the decisions are taken at a local level, i.e. in the junction controller [4].

In [4] the performance of these algorithms have been tested against a centralised one, the centralized approach have obtained better results during the performed tests. Despite this results, looking for decentralized solutions is still interesting, the decentralized solutions are good avoiding escalation problems that usually arise in the centralized environments where high amounts of data must be managed. Besides, these solutions may be useful at a fine grain level, improving in this way, the network efficiency with a low investment [4]

1.6 Genetic Algorithms (GA)

GA have certain advantages in optimization problems such as the one I am facing in this paper. GA were designed to resemble the way in that living beings evolves in nature,furthermore, the genetic principles that rule the species evolution are computationally mimicked. This approach has shown being very useful managing optimization problems [5], some of these advantages are:

1. The great benefit of fitness function if we compare it against other optimization solutions is that the fitness function is highly adaptive, It can be mathematically defined to be what we want to be, this give us versatility in every situation, basically, this means that we are using coded versions rather than the parameters [5], then, we can tweak it in our benefit.

2. GA are also very good in finding the global optimum solution because they use a population of solutions rather than a single solution,as other optimization techniques do [5], in this way, We can explore solutions that other methods couldn't.

3. GA can be applied to either continuous or discrete problems because they use fitness functions rather than derivatives, because of this, it is very important defining correctly the function that interprets all this information [5].

4. GA doesn't use deterministic rules, i.e the transitions are probabilistic, so they are suitable for non-determinist environments, this is a major advantage, since it provides versatility.

Chapter 2

State of the Art

This chapter is focused on taking a glance to novel proposals related with traffic management. Most of them follow an IA approach, however, I have also included other which don't.

2.1 Genetic Algorithm Solutions

Along this section, I am going to show some solutions proposed in the field of traffic management using genetic algorithms.

2.1.1 Optimal Control for Traffic Flows in the Urban Road Networks and Its Solution by Variational Genetic Algorithm

In [6] the authors have proposed a solution using genetic algorithms yielding small variations in the basic control schedule. An ordered set of small variations is implemented at the beginning of the experiment. Each individual is a tuple containing, the index of the intersection, the control time index an a third parameter that stands for control program which can take boolean values. The whole population of variations is applied to the basic solution and then a fitness parameter is computed.

Each variation set i:

$$V_i = [v_1, v_2, v_3] \tag{2.1}$$

Population of variations:

$$W_i = [V_1, V_2, \cdots, V_i] \tag{2.2}$$

Apply the set of variations to the basic control program

$$W_i \circ \hat{P}(\cdot) = W_1 \circ \hat{P_1}(\cdot), W_2 \circ \hat{P_2}(\cdot), W_i \circ \hat{P_i}(\cdot) \tag{2.3}$$

As usual, in order to guarantee a fair search into the solutions space, mutation is used.

2.1.2 Active control for traffic lights in regions and corridors: an approach based on evolutionary computation

In the case of [7], the solution is based on the utilization of an evolutionary algorithm. The proposed algorithm takes parameters from the environment with the objective of build a mathematical model which estimates the average delay time of vehicles in the whole network. This information is used to feed an evolutionary genetic algorithm, which is employed to update the green time in all the possible movements into the network at a fine-grained level.
The proposed fitness function is:

$$f_f(0) = \frac{1}{\overrightarrow{x} + k_p \cdot f_p(\overrightarrow{x})} \tag{2.4}$$

Where $\overrightarrow{x}$ is the green time vector, k_p is a penalty constant, f_o and f_p are the objective and penalty function respectively.
The objective function is those that minimizes the average delay time and represents the difference between the time taken by a vehicle to travel through a intersection geared by TL whit respect to a situation of free flow. The delay time expression is:

$$f_d(0) = \frac{(c \cdot (1 - p(\overrightarrow{x}))}{2 \cdot (1 - p(\overrightarrow{x})) \cdot s} \tag{2.5}$$

Here the parameter c corresponds with the cycle length, p is the fraction of green time with respect to the cycle, i.e. $\frac{p(x_i)}{c}$, and s is the degree of saturation obtained from divides the traffic demand by the capacity.
Several combinations os the selection method and mutation had been tested, with low differences between them but obtaining a decrease of 47% with respect to the actuated solution in use at that time in the network.

2.2 Reinforcement Learning Solutions

Along this section I am going to explore some solutions based on reinforcement learning, which is an approach typically used to face these kind of problems.

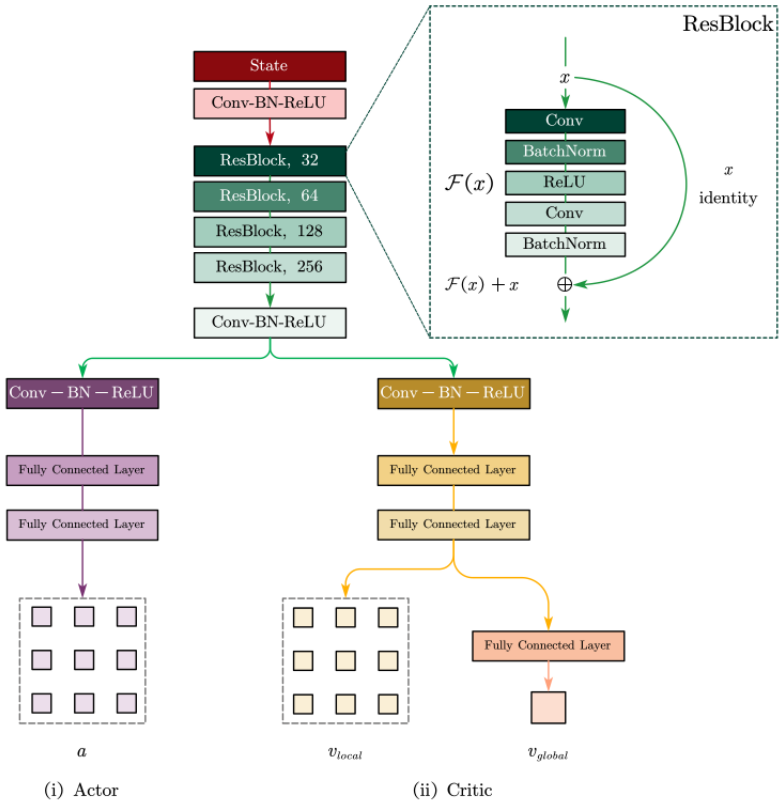
2.2.1 An Efficient Deep Reinforcement Learning Model for Urban Traffic Control

In the paper [8] an interesting approach is taken, the authors propose apply a deep neural network, modelling the traffic agent as an Advantage Actor-Critic, as we see in the image 2.1

In my opinion, the more interesting of this solution if that they leverage techniques mainly used in other fields, as the DNN, largely used in computer vision. They conceive each TL intersection as a matrix which is combined with every matrices of the network, shaping a grid filled with parameters, each intersection is formed by eight tensors monitoring the halting vehicles on each lane, where input and output lanes are both considered 2.2. The reward function is obtained as an average of the vehicles entering and leaving the grid, another reward function is computed at a local intersection level and corresponds with negative difference between queue length in the lanes with direction north-south, south-north and the lanes where direction is east-west or west-east.
The function of the critic DNN is computing the value function, while the actor DNN must update the policy taking into account the parameter returned by the critic. In the current paper, another element is added to the model, in this case, a the upper layer are shared between the actor and the critic, which seems to yield a better performance.

Once trained, the network should be able to return a list which maximizes the reward, following in this way an optimal policy, the parameter is a set of boolean elements, and will be either keep the current stage or change it, this value is issued for all the TL of the grid. The action space, that is to say, the state space is defined by the stable phases that can be applied on each TL junction.

FIGURE 2.1: Advantage Actor-Critic Model [8]



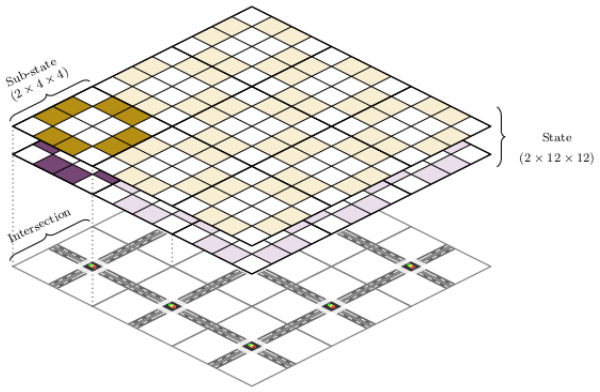
(i) Actor (ii) Critic

2.2.2 A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management

The authors of the paper [9] propose a solution where the state space is defined by the position of the vehicles in an instant t ,to achieve such a definition of state spaces, they have designed a mechanism where the approaches to the junction are divided in cells, and its length is greater as we move away the junction. This is motivated to avoid losing vehicles information and avoiding a excessively large state space. As in the previous work the action space is a set of fixed plans. The algorithm are based in two type of rewards, the total waiting time of the vehicles in the intersection, and the accumulated total waiting time,measured both for an agent step, the different between them stands in the fact the first reset the time of a vehicle for each step, the second parameter , keep increasing its value until the vehicle crosses the intersection, then, a reduction of these parameters must be the objective of the training process.

The training is performed using an experience replay buffer, where the training data is delivered to the agent in form of a set of selected at random samples get from the replay buffer. The data consists on a set of 80 cell values at a certain instant t which are used to feed a NN with 5 hidden layers with 400 neurons each.

FIGURE 2.2: Grid Architecture Proposed in[8]



2.3 No AI Based Solutions

2.3.1 The Job Scheduling Algorithm

In this algorithm the TL environment is considered as a scheduling problem, actually, it is. The vehicles are grouped into clusters, moving at constant speed toward the stop line. The algorithms studies all possible combinations to obtain a schedule suitable to serve the current clusters. Although this algorithm has obtained improvements in real scenarios has also shown problem with respect to computation time (in certain scenarios)[4] which increases exponentially when the number of vehicle cluster grows.

2.3.2 The Max-Pressure Algorithm

The idea behind this algorithm is taken from the communications network control and stems from the scheduling algorithms gearing it. The vehicles are considered customers, in the same way are the data flows considered in a network infrastructure, or clients in client server distributed system. As we know, a network has constrained resources and not all clients can be served simultaneously, in the same way, a TL environment can not serve preference to all vehicles simultaneously. In a junction, switching from a state to another (phase changing), introduces lost times, these lost times comes from transition phase times, so, switching between phases many times increases this accumulated lost time. In order to solve this problem,that doesn't exist in the network protocols, a solution has been explored, which consist in defining minimum and maximum green times, also respecting the stage sequence, avoiding in this way confusions in drivers [4]. The algorithm measures the pressure of each approach to the junction, this is obtained using the waiting queue length or the number of vehicles on each junction approach,the control decision consist on a time extension, (2 sec for example) , always that doesn't exceed the time boundary limit, and at the same time, it is observed that the initial measured pressure has not been reduced by more than defined percentage or it is the maximum-pressure stage,if these conditions are not met the next phase in the sequence is activated.

2.4 Limitations

All these proposed solutions, in my opinion, have some issues or constraints, for example, the solutions based on GA, despite the fitness parameters selected seem appropriate, both are centralized solutions, they are designed to train algorithms for the whole network, this fact, in my opinion, reduces the resilience, because, if a node losses communication with the control center, couldn't receive orders from there. Furthermore, they both loss the option to be deployed in those places where building a centralized system was excessively expensive. Besides, I consider that both are little scalable, the length of the chromosomes of the GA might become very large in scenarios with many junctions, increasing the search space exponentially and making both options impracticable.

On the other hand, the reinforcement learning solution based on actor critic, although, I reckon that it is conceptually awesome but, has a problem of adaptability. Training such a type of models is very slow but, it is true that, once trained, the model could work fine, in the case of a change in the flows,due to, for example, a street closed for repairs, or just a change in the drivers habits, might make the network collapses . I believe that training the system to such a variety of situations is not easy, and must be carefully studied in order to handle unexpected events. The issues that I observe in [2] can be applied to the second reinforcement learning solution.

In the case of no IA based solutions, as I had commented, [4] has a scalability problem due to exponential growth of computational costs. The max-pressure algorithm, is in my opinion the most interesting algorithm from the studied here, for example, it implements an upper and lower limit to the green light time and also look for a fair distribution of cycle time, otherwise, I have doubts about its behaviour in relaxed conditions where, even in this case , the performance might be optimized.

Chapter 3

Genetic Algorithm Traffic Agent

3.1 Introduction

In this chapter I am going to present the key aspects of the solution proposed. First, I will explore the architecture of the system, then, step by step I am going to analyse its different parts more in detail. This work is focused on the logic that gear the agent solution, so the rest of the system is added merely to put the reader into context and we will not do a deep study about it.

3.2 Proposed Solution

As we have seen, the decentralization in UTC has some strengths, by itself or merged with a centralized approach,so, It would be interesting to look for solutions that leverage an decentralized idea. We have also seen the advantages of using GA in order to solve optimization problems in non-deterministic environments as the one. That is why I am going to explore a combination of this two approaches, decentralization and GA. Next, I will briefly speak about the elements in which my solution will be based.

3.2.1 The traffic controller

The traffic controller is the "brain" of a TL controlled junction. It is an automated system equipped with all the necessary electronic instrumentation to carry out its mission. Today, these controllers are powerful computers with advanced technology capable of perform highly demanding tasks.The actions taken by the controller are used to activate the traffic lights in function of the parameters collected by it. My proposed algorithm will "live" into a device of this type.

3.2.2 The environment

Sticking to the Cambridge digital dictionary definition says, we can see one of the entries which I find very nice to illustrate my case, We can read that the environment is: "the situation that you live or work in, and how it changes how you feel" [10]. If we transfer this concept to our TL world, we can understand it as the place where our controller works, that is to say, the junction where it is placed, doing its work, i.e. changing the junction stages and changing the way in that it feels in function of the measures that it obtains from the sensors,pedestrian buttons or other elements installed in the proximity that act as its senses, or even, external information received from other controllers or control room facilities.

3.2.3 The optimization problem

We know that GA are very good at solving optimization problems, we are aware of the fact that the traffic load in a junction depends on many parameters and can vary very fast. We also know that there is a certain risk of reaching a congestion state when some conditions are met. To avoid the congestion state, and providing a fair time distribution between approaches to the junction the green time assigned must be balanced in a suitable way. Then, we can define an optimization problem where we are going to find the best time distribution for the TL for the current traffic load. The time assigned to each stage of the cycle will correspond to a gene value, and the fitness function will depend on parameters taken from the junction environment in a isolated way, the traffic controller taking decisions by its own strengthen the decentralized nature of the solution.

3.2.4 The simulator

Since we are working in an environment where doing tests against the real one might have serious implications (may cause congestion or even accidents), it is a good idea to train and test our algorithm against a simulator. There are multiple traffic simulators, some are free software, as the one I have chosen for my experiments. The SUMO Simulator [11] is an open source traffic simulation solution quite powerful and it is perfect for my purposes,furthermore, it is important to note that SUMO provides a tool to get information from the simulation known as Traci, this solution has an interface that let us connect with the simulation using python, this will be very useful in connecting our GA with the simulation .

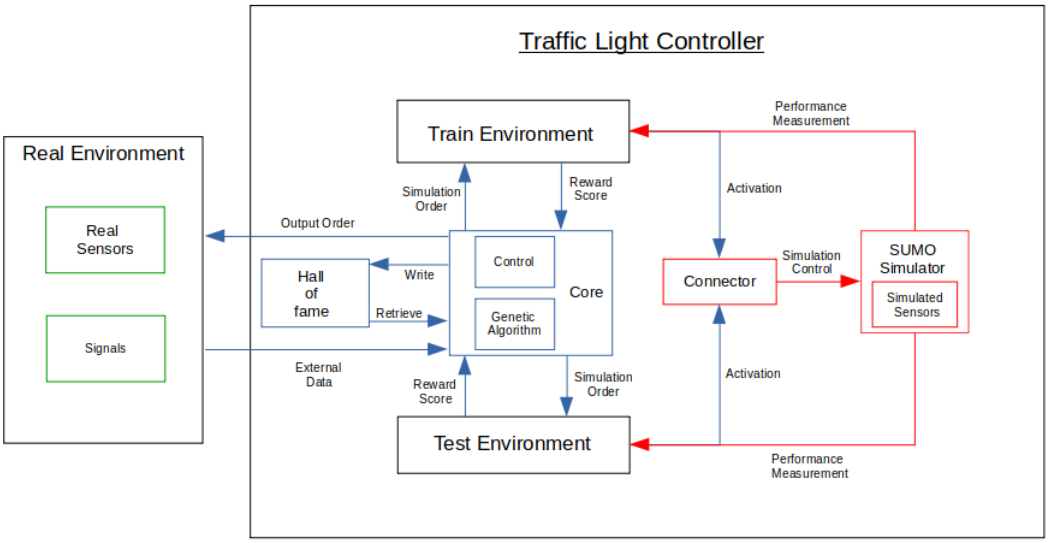
3.2.5 A traffic agent based on genetic algorithms

The solution consist on a traffic agent based on GA. The algorithm is built using the DEAP python framework [12],which is published under the LGPL license and it is a very versatile tool for testing purposes in relation with evolutionary algorithms. The concept rest,as we have spoken, on the interaction between the GA,the environment and the simulator. In the environment side I decided to use tensor-flow, specifically, tf-agents [13], this framework is structured in a way that let us the deployment of environments very quickly, although it was designed to be used along with reinforcement learning algorithms, it is easily adaptable and can be leveraged for the concept presented in this work.

3.3 The architecture

The architecture 3.1 is built in two main parts, the traffic light controller and the real environment. Into the traffic light controller resides all the program logic, the simulation, the train/test environments and the GA. The real environment is used to apply the solutions found by the algorithm and to collect real data, but this part and its interconnection with the core algorithm , as I have previously stated, will not be analysed.

FIGURE 3.1: Traffic Agent Architecture [2]



3.3.1 Traffic Light Controller

The proposed solution should run into a traffic light controller device. This device should have the computation power to run the mathematical computations as effective and fast as possible with a high degree of reliability. The different parts that shape the algorithm are:

Core

The core part is subdivided in other two parts, the control and the genetic algorithm parts. The control is the program logic that interacts with the hardware of the controller and might eventually send and receive orders from it. The other part , much more related with this work, is the GA logic which is in charge of the optimization problem and must interact with the simulations and other elements of the architecture.

Train environment

The train environment is an instance of the environment class used to evolve the individuals into chromosome's population towards the best possible. This element must interact with the simulator and returns a reward in function of the performance of each individual. The returned reward value is afterwards used by the fitness function to obtain a suitable evaluation.

Test Environment

This environment instance is included to test the non dominated individuals obtained after each training process. Hypothetically, the individual with the best performance against the test environment might be applied to the real scenario. It is important to note that, both, test environment and train environment, must resemble

the most possible the real one, this has sense, because the obtained optimal solution should be after, applied to a real scenario.

Connector

This code piece is used, as its name says, to connect the environments with the simulator, it is in charge of beginning a simulation, sending step orders and finishing it.

Simulator

The simulator is in this paper an external tool used to simulate the real environment,as I said, it is SUMO simulator. SUMO allows parallel simulation execution and provides a powerful interface that let us read or manipulate parameters online, this is very interesting because we are going to test the performance of the population, and we need to apply it to the simulation, and then measuring its response.

Hall of fame (HOF)

The HOF is a variable where the best individuals are stored, this list is mutable, since the best individuals will change along time, this happens because the traffic evolves, causing that the time distribution can't be always the same. An individual can be a very good solution when there is traffic congestion in a certain approach to the junction and being a very bad solution when this same approach has a state of free flow. Because of this the GA must actualize the HOF when needed.

3.3.2 Real environment

The real environment corresponds to the echo-system where the TL controller is installed, it is divided in two parts the traffic light elements and the sensors. The sensors are all the elements that lets the controller interact with the outside world, as induction loops, cameras doted with artificial visions systems, pedestrian push buttons, orders from a control center and so on, doing an analogy with the human beings are the senses of the controller. On the other hand the traffic lights or signals are the elements that translates the controller intentions toward the outside world, following the former analogy are the limbs of the controller. As we said, actually, in this stage, I am not very interested in this part of the system, for this research phase, we won't need to attach the algorithm to a real environment, since its changes are emulated in the ad-hoc tool, which will let us avoid performing tests against a real scenario, which may have serious implications. However, it is necessary locate it into the architecture for hypothetical further research.

3.4 Definition of the problem

We must find the best plan for a TL junction, the plan will be defined by the duration of the stable stages and an offset parameter,which will be an action applied to the simulation environment. The objective is mapping from a environment lecture of parameters the best suited chromosome to that condition. The environmental parameter that I am using is the jam length, that is to say, the average length of the queue in an approach to the junction. This is measured in SUMO using a lane area detector, which mimics the way in that an artificial vision camera works. We can

retrieve from the simulation interesting information, as the mentioned, that is to say, the instant jam length, but we might eventually retrieve, for example, the instant occupancy. Note that, this gauge infrastructure doesn't have to be implemented in the real environment,since the jam length is only necessary during the simulation process,so, much simpler mechanisms would be enough.

The best plan will be a vector containing certain natural numbers in a range, which will correspond with the time to apply to each stage and an offset parameter which takes values in a range into the set of the integer numbers. So, we can define the problem mathematically as finding the optimal solution for function:

$$g : \mathbb{R}^n \longrightarrow (Y^n, O) \tag{3.1}$$

where:

$$\text{n=}\textit{number of stable stages}$$

with:

$$Y \subseteq \{\mathbb{N} : [\textit{min stable stage time}, \textit{max stable stage time}]\} \tag{3.2}$$

$$O \subseteq \{\mathbb{Z} : [\textit{min offset time}, \textit{max offset time}]\} \tag{3.3}$$

The lower and upper duration stage limits are selected according to the scenario. The minimum time must be enough to guarantee the time necessary to pedestrians for crossing the street during the related stage, and the maximum value can't be excessively high, because, we don't want the users spending unreasonable time waiting on its approach.

3.5 Initial population

The initial chromosome's population is created using a pseudorandom number generator, the obtained integer values are constrained to the lower and upper time bounds established. The generator receives a set with the integers between the lower and upper limits and returns a vector with length equals to the number of stable stages, these values are randomly selected following a discrete and uniform probability distribution. This process is repeated until had been created the number of individuals desired.

$$(low, low+1, \ldots, up-1, up) \xrightarrow{U} (time_1, time_2, \ldots, time_i) \tag{3.4}$$

Where $(time_1, time_2, \ldots, time_i)$ is a multidimensional discrete random variable and $time_i$ is a discrete random variable such that:

$$time \sim U(low, up) \tag{3.5}$$

3.6 The fitness function

The fitness function is used to evaluate the performance of an individual taken from the population for the current environment state. For this problem, I have designed a function which takes as input the jam length vector, and outputs another vector with three elements, the first one is the mean value of the input vector of jams, the second element is the standard deviation obtained from input vector and the third

one is the absolute value of the sum of the vector gradient elements obtained from the input vector, the latter value is closer to zero if the jam length values are closer one of another.

$$f : (Y^n, O) \longrightarrow \mathbb{R}^3 \tag{3.6}$$

$$f(x_0, x_1, \cdots, x_{n-1}) = \left(\frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i, \sqrt{\frac{1}{n} \cdot \sum_{i=0}^{n-1} (x_i - \mu)^2}, \left| \sum \overrightarrow{\nabla} \overrightarrow{x} \right| \right) \tag{3.7}$$

The obejective is to find the chromosome that returns the min fitness vector, i.e. those which returns the minimum combination of the *rounded* values from the three functions.

$$f_{min}(x_0, x_1, \cdots, x_{n-1}) = min \left\lfloor \left(\frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i, \sqrt{\frac{1}{n} \cdot \sum_{i=0}^{n-1} (x_i - \mu)^2}, \left| \sum \overrightarrow{\nabla} \overrightarrow{x} \right| \right) \right\rfloor \tag{3.8}$$

where:

$$\overrightarrow{\nabla} \overrightarrow{x} = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_{n-1}} \right) \tag{3.9}$$

Minimizing the mean I will try to reduce the queue on each approach. Minimizing the standard deviation and the sum of the absolute value of sum of gradient components I will favour a fair queue distribution between the approaches, the latter may be considered a balance parameter. This perspective should assure that the time will be reduced in those approaches with less traffic,although the jam length may increase. Actually, never mind, that the queue increases slightly in an approach with scant traffic, because, if we focus only in the mean, we might reduce excessively the jam in one approach where wasn't necessary and don't in other where actually it is, getting a worse performance result, although, hypothetically, the mean might be better.

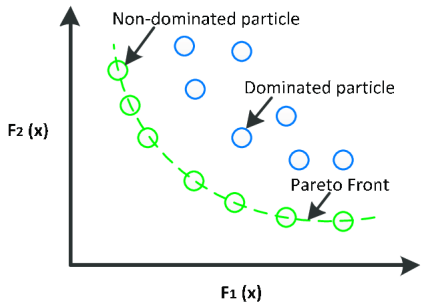
3.7 Hall of fame (HOF) logic

In the hall of fame we are going to store the best individuals along the process. Each iteration the non-dominated individuals are stored into it. The hall of fame principle used is Pareto front,the front is formed by the non-dominated elements in the analysis scope, the individuals which pertain to this group will fill the Pareto set. This individuals, in my proposed algorithm, are stored in the HOF. The Pareto front concept was defined to solve multi objective optimization problems [14] , here, the objective, is finding the non dominated solutions into the functions space. A solution is considered optimal and therefore belonging to Pareto front if no objective function can be improved without sacrificing another objective function . On the other hand, a solution is considered dominated by other if the other solution is as good as,or better in all objective functions.

This is our case, we have three different functions and we are trying to find the optimal solution for them, in our case, the non dominated optimal solutions which minimizes the three functions will be stored in the HOF. After the completion of a round all the chromosomes stored in the HOF are tested against the test environment

and its fitness value is updated when the environment parameters are modified. Doing that, I am trying to increase the population performance between the two scenarios, this is very important because the optimal solution should be applied in a third random environment, so we need individuals that perform well against all of them.

FIGURE 3.2: Pareto Front Plot [15]



3.8 Crossover

The crossover is involved in the process by means a new population is derived from the previous one. It consist on swapping the genes from one individual with other and vice versa, in my design this happens from one point onwards. This action assures that the algorithm explores a larger solutions space. The offspring obtained by this method is given by a crossover probability.

I am going to show the crossover process with an example where are given the parent individuals:

$$\overrightarrow{a} = (a_0, a_1, a_2, a_3, a_4) \tag{3.10}$$

$$\overrightarrow{b} = (b_0, b_1, b_2, b_3, b_4) \tag{3.11}$$

The offspring obtained if we swap from point 2 is:

$$\overrightarrow{c} = (a_0, a_1, b_2, b_3, b_4) \tag{3.12}$$

$$\overrightarrow{d} = (b_0, b_1, a_2, a_3, a_4) \tag{3.13}$$

Note that the swapping happens with a $prob_{crossover}$ and the crossover index point,which in the case of our example is two, is selected with uniform probability.

3.9 Mutation

It is also interesting increase the exploration space performing a mutation in some individuals of the current generation. In my algorithm the mutation is carried out with a certain probability $prob_{mutate}$. In the approach used in the current solution if an individual is selected to be mutated each of its genes are mutated with probability $\frac{1}{chromosome\ length}$, the mutation is performed generating random at uniform integer value in the range $[min\ stable\ stage\ time, max\ stable\ stage\ time]$. As I did with the crossover process i will try to illustrate the mutation with an example.

Given the individual selected to be mutated:

$$\overrightarrow{a} = (a_0, a_1, b_2, b_3, b_4) \tag{3.14}$$

A possible mutation might be:

$$\overrightarrow{b} = (a_0, c_1, b_2, b_3, c_4) \tag{3.15}$$

Where the c_1 and c_4 are integers random uniformly selected respectively in the range:
$[min\ stable\ stage\ time, max\ stable\ stage\ time] \& [min\ offset\ time, max\ offset\ time]$

3.10 Selection

The selection process is performed on the population living in the current generation, its objective is selecting those individuals with a better fitness to be used for breeding in the next generation. The algorithm used for the selection process is NSGA-II [16].

This algorithm was developed to solve multi objective optimization problems, this fact makes it suitable for selection in the current TL problem. NSGA-II works following the next steps [16]:

- The algorithm takes the current parents population P_t.
- Then, generates a offspring Q_t from the parents set P_t using crossover and mutation.
- A new combined population $R_t = P_t \cup Q_t$ is generated.
- This just created population is sorted using a crowding distance sorting approach. This is done using a crowded comparison operator, this operator leverages a density estimation, which build a cuboid surrounding each solution,created taking the points at each side, then, the crowding distance will be the average of the sides of the former cuboid, that we can see in 3.3, this process is repeated for all objective functions, the sum of all these values is the overall crowding distance.

  Then, the solutions with smaller overall distance are supposed to be more crowded by other. So, selecting the less crowded solutions we can preserve the diversity.
- The solutions are sorted in several tiers, that is to say, several Pareto front sets are created, $F_1, F_2, \ldots$.
- Now, if the size of F_1 is less that the number N of elements in Q_t the algorithm selects all elements of F_1 for the next P_{t+1} parents generation.
- In that case, the remaining parents population are selected in order,with preference to elements in set F_2 and so on, until the number N of elements is filled, this process is illustrated in 3.4.

FIGURE 3.3: Cuboid Example[15]

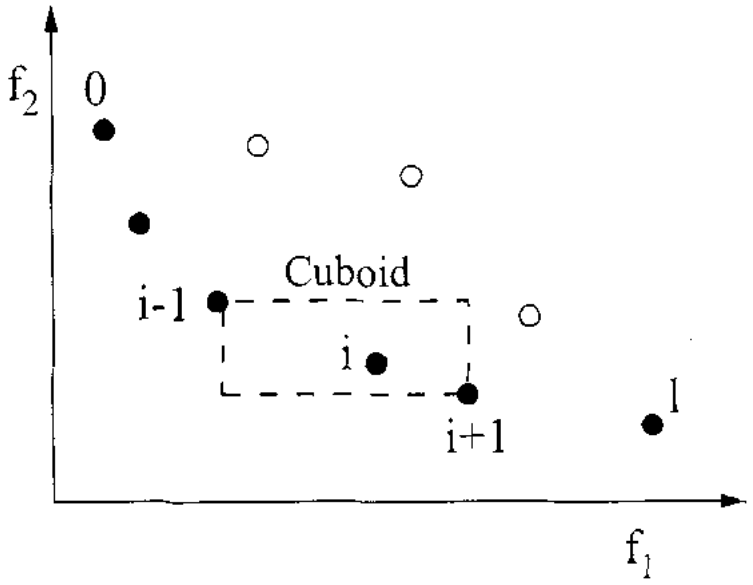
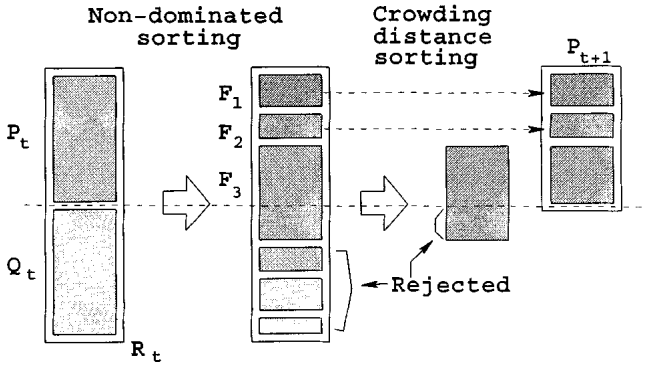


FIGURE 3.4: Selection Procedure in NSGA-II[15]



3.11 Algorithm

All the previous techniques are combined in a single algorithm with the aim of solving the traffic problem supported by the TL architecture. The algorithm which summarizes the operation is depicted in algorithm 1.

It is also interesting showing the score function algorithm used to compute the individual score, this function is the core of the system since it is used to measure the individual performance, we can see it in algorithm 2.

Algorithm 1: Genetic Algorithm Traffic Agent

```
Result: HOF With Optimal Plans
DefineParameters;
GetParameters(configuration files);
InitsimulationEnvironments;
CreateInitialRandomPopulation(population length, individual length,
  bounds);
Define toolbox elements;
Init eaMuPlusAlgorithm;
for generation ← 1 to number of generations do
    if generation==number of generations then
        for i ← 1 to population length do
            score ← Score(individual);
            if score is non − dominated then
                AddtoHof(individual);
            end
        end
        newhalloffame ← TestHofIndividuals(hall of fame);
        bestIndividual ← SelectNonDominated(newhalloffame);
        ComputeVolumetoCapacityRatio(newhalloffame);
        Continue to next round;
    else
        for i ← 1 to population length do
            score ← Score(individual);
            if individual is non − dominated then
                AddtoHof(individual);
            end
        end
        selected ← Selection(number of individuals to select);
        for i ← 1 to number of childrens do
            Crossover(crossover probability, selected);
            Mutation(mutation probability, selected);
        end
    end
end
```

Algorithm 2: Score Algorithm (Fitness Function)

```
Result: Individual Scores List
SetOffset(traffic lights file, individual offset);
for counter ← 1 to simulation steps do
    OrderSimulatorStep(individual);
    timestep ← GetTimeStepVariableFromEnvironment(individual);
    if step is last step then
        break;
    end
end
mean ← ComputeMean(timestep.reward);
std ← ComputeStandardDeviation(timestep.reward);
divergence ← ComputeDivergence(timestep.reward);
individualscoreslist ← round(mean ,std ,divergence);
```

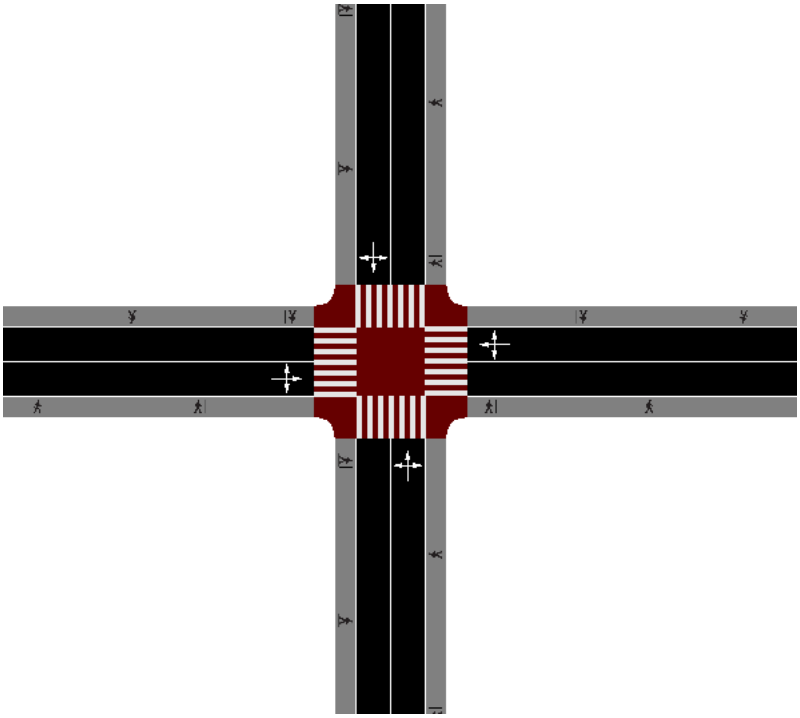
Chapter 4

Experimental Results

4.1 Introduction

Testing the performance of the solution will be the objective of the current section. The solution will be simulated in an environment with four stable stages where all the movements are allowed, i.e., at each approach to the junction the driver can drive toward the left,toward the right or continue straight as we see in 4.1.

FIGURE 4.1: Experimental Intersection



This intersection structure was selected looking for a difficult scenario where was hard to accommodate all the traffic flows. I have configured independent stages per incoming lanes, so may be hard find an optimal solution. As we can see, there is a single incoming lane to the junction per approach, where all movements are allowed. Besides, there are four pedestrian crossing.

There are also included in the simulation four lane area detectors with 150 meters length, used to measure the jam length. In the image 4.2 we can see in blue color the detector, we can also see coloured in light green the allowed movements during the first stage and in red the not allowed, this corresponds with the TL logic scheduled. The dark green are allowed movements with no preference, in this case the driver must yield at the pedestrians,and it works as an intermittent yellow light

FIGURE 4.2: Experimental Intersection

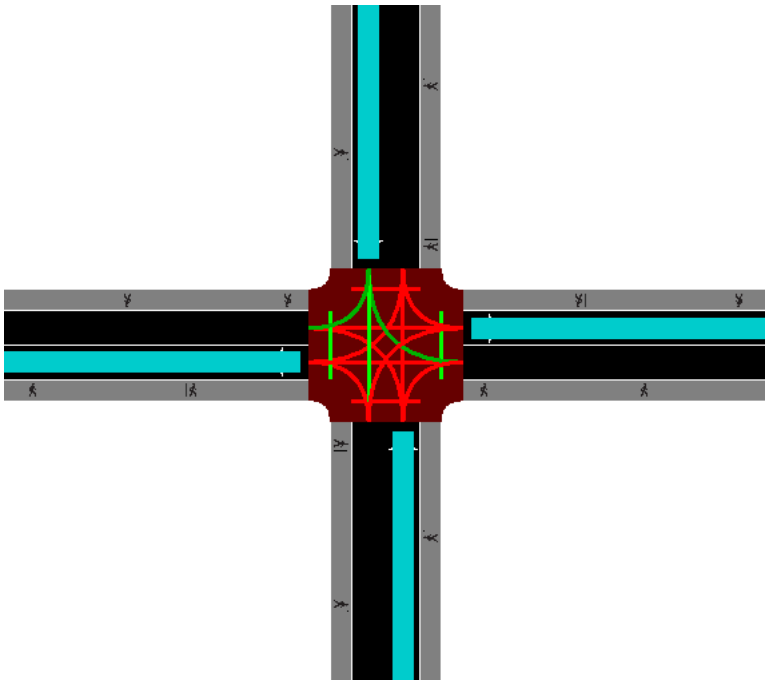


TABLE 4.1: Stable Stages

Stage	Location	Order
SP1 (Stable Stage 1)	North Lane	1
SP2 (Stable Stage 2)	West Lane	2
SP3 (Stable Stage 3)	East Lane	3
SP4 (Stable Stage 4)	South Lane	4

For each lane, one flow is simulated for each movement, as is depicted in 4.3, I also simulate the pedestrian flows, both, vehicles and pedestrians are issued randomly according to a certain probability per each step simulated, which would correspond with a second.

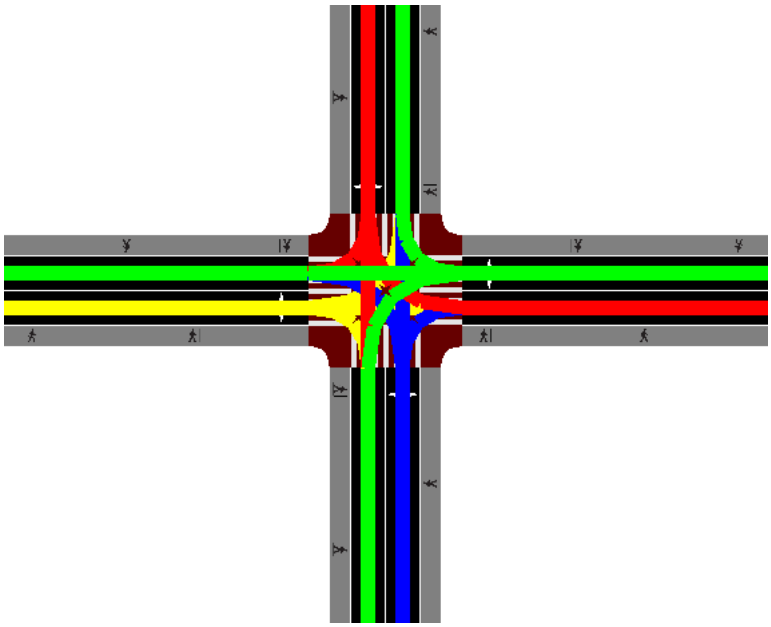
The SUMO simulator is used to generate all the logic of the simulation, several tools are provided along it which can be used to perform this tasks, the images 4.2 and 4.1 have been obtained using it, I delve into this tools in C. This tools generate a bunch of xml files used afterwards by the designed software to interact with the simulation,letting us getting measures and to change its parameters dynamically.

4.2 Traffic lights logic

The logic defines the structure of the TL sequence, i.e., where the stable stages and the transitory stages are placed, in my design, the stable phases (stages) are numbered according to 4.1.

There are two positions for each transitory phase when changing between stable phases. All the logic is resumed in 4.4. Each group controls the pass for a group of movements, for example, in 4.2 we see the group 1, the group 6 and the group 7 showing green state, and the rest of groups are showing red state. When we are

FIGURE 4.3: Experimental Intersection with Flows



handling the timing we are changing the time allocated for each stable phase, this let us control the number of cars passing the intersection on each cycle.

In 4.3 the vehicle flows are shown, there are three flows per lane, letting us define different load parameters for each movement into the intersection.

4.3 Simulation parameters

The parameters have been selected following a empirical approach, the length of the initial population, that is to say, the population size parameter, must be long enough to guarantee a broad space exploration, the mu and lambda parameters selected have shown good performance for this environment, this also applies for others, as the crossover probability or mutation probability, in my experiments setting the values as the depicted below have shown good performance. A very low value in the crossover parameter have shown that the algorithm is slower in finding an optimal solution with slight performance improvement, and a excessively high mutation probability made it quite unstable. Parameter selection is highly tied to the scenario where the algorithm is executed and must be selected accordingly.

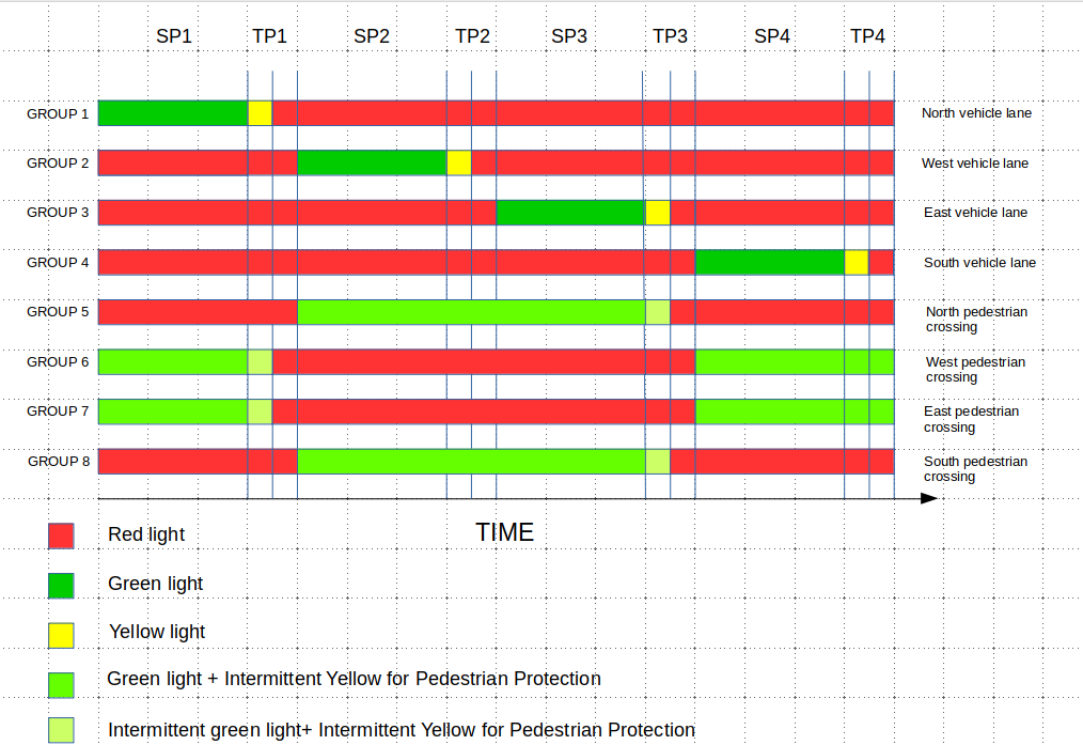
All provided values below are fixed during the different experiments performed but the number of generations which is tuned to different values in function of the simulation type, being this, simple or evolutionary.

- **Number of phases:** 4
  This value is get from the net configuration files, in this case the number of phases is 4.

- **Chromosome length:** $number\ of\ phases + 1$
  This value is 5 in my experiment and corresponds with the number of stable phases plus the offset parameter.

- **Population size:** 150
  This value states the initial population set length.

FIGURE 4.4: Intersection TL timing



- **Mu:** 30
  With mu I am defining the number of individuals to be selected for the next generation.

- **Lambda:** 50
  The offspring set must have a length equals to the lambda parameter.

- **Crossover probability:** 0.8
  Probability of an individual being selected for crossover.

- **Mutation probability:** 0.2
  Probability of an individual being selected for mutation.

- **Max generations:** 5
  The number of generations computed.

- **Number of steps:** 450
  The number of steps carried out by the simulator, a step corresponds with a second and it is defined in the sumo configuration files.

- **Cycle:** 120
  This parameter is used when we adapt the phases to a fixed cycle value and to compute the maximum phase time.

- **Minimum phase time:** 15
  Lower bound for phase time

- **Maximum phase time:** $Cycle - (min\ phase\ time \times number\ of\ phases)$
  This value is obtained using the cycle defined and the number of phases

TABLE 4.2: Vehicle flows

Probability	From	To
0.04	North	East
0.04	North	South
0.04	North	West
0.01	West	East
0.01	West	North
0.01	West	South
0.02	East	North
0.02	East	South
0.02	East	West
0.03	South	East
0.03	South	North
0.03	South	West

- **Minimum offset time:** 0
  The minimum offset value is set to zero, in this case no offset is applied.

- **Maximum offset time:** 100
  Max offset value parameter.

- **Random seed:** Integer randomly selected between 0 and 100
  Changing this value guarantees randomness in our experiments.

- **Number of experiments:** 5
  When the experiments are performed using the test file, that is to say,changing the traffic flow vehicle loaded during the execution, this parameter states the number of experiments to perform.

4.4 Results

4.4.1 Results in a simple scenario

The first experiment will be performed using the flow parameters reflected in tables 4.2 for vehicles and in 4.3 for pedestrians. In both cases the probability value defines the probability to generate a car or a pedestrian in a certain second. Each flow has a certain route defined as the from/to and origin/destiny parameters. Setting the duration of the simulation in 450 steps we obtain the plan performance for the short term. The random seed is renewed each time the script is executed, the sumo simulator is also initiated with different seed parameter each time is executed, this fact, increases the difficulty.

The first execution is performed using the gatraffic.py program, where I simulate a single epoch of the algorithm,that is to say, a single experiment,therefore, maintaining fixed the flow probability parameter. All the parameters are taken from the values shown in section 4.3 obtaining the following results, summarized in table 4.4:

The train evolution is obtained for the mean of the fitness score parameters for all the population along the 5 generations. I have repeated the experiment with the previous parameters and changing the number of generations to 25 and 40. The

TABLE 4.3: Pedestrian flows

Probability	Origin	Destiny
0.1	East	West
0.1	North	South
0.1	South	North
0.1	West	East

TABLE 4.4: Experiment 1 in simple scenario

	5 generations	25 generations	40 generations
Duration	450	450	450
Intensity	[432,108, 216, 324]	[432, 108, 216, 324]	[432, 108, 216, 324]
Best Time Schedule	[48, 31, 18, 50, 15]	[40, 18, 25, 35, 27]	[60, 18, 23, 38, 45]
Best v2c Time Schedule	[50, 23, 33, 42, 17]	[58, 17, 49, 45, 15]	[57, 18, 25, 42, 41]
v2c value	0.657	0.645	0.654
Execution time	192 sec.	698 sec.	1208 sec.

results are shown in the panel plot 4.5 . With respect to the jam length, I am using the Moving Average 50 (MA50) statistical tool, MA50 is broadly used in economics to analyze the evolution of stock values. Its use in our scope will smooth the jam length evolution and, in this way, we can appreciate more clearly the evolution, letting us evaluate the jam length from generation to generation for the four lanes defined in the experiment.

As we see in the results, although, the plots depict an evolution towards better mean score values, when I increase the number of iterations (generations) the mean volume to capacity ratio has a very similar value in all of them. This happens because the algorithm is designed to split the jam between all the lanes, so when this value is improved in a lane may be ,otherwise, worsened in other,besides, the objective is keeping a good volume to capacity ratio possible in function of the environment constraints, so, in many circumstances, reducing this parameter will not be possible.

The volume to capacity ratio in a TL environment is computed as follows [17]:

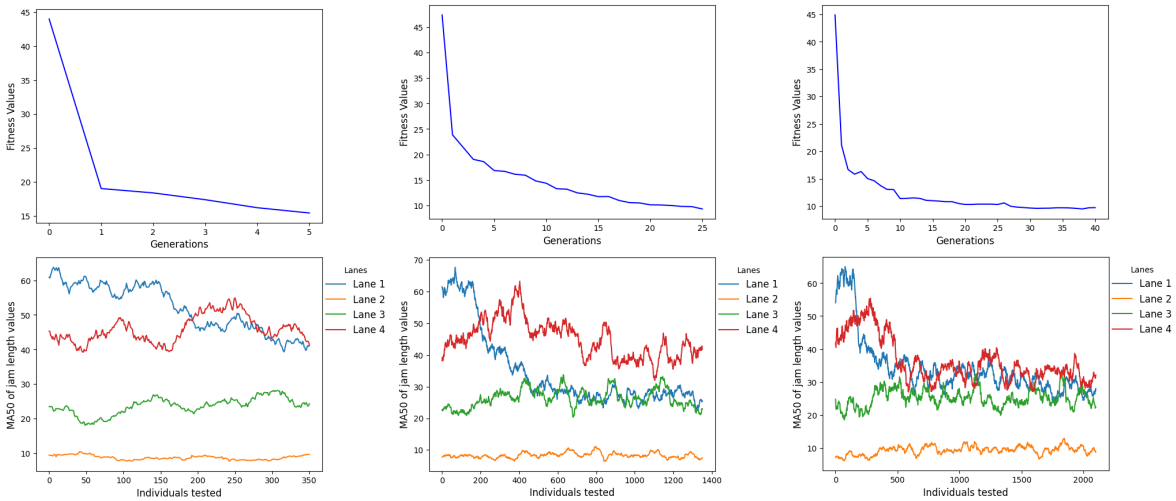
$$VC_r = \frac{N_p}{N_{max}} \tag{4.1}$$

$$N_{max} = S \times \frac{V}{T} \tag{4.2}$$

In this equation S is the saturation intensity in veh/h, V is the duration of the green phase for the lane and T is the cycle duration. This equation is slightly different to equation 1.6 ,although both are used to compute the same parameter,this is because in the TL environment, the movements are not always allowed, so, this regard, must be taken into account. The saturation parameter is obtained from the following equation [17]

$$S = 1900 \times N_l \times f_w f_{tv} f_i f_p f_{bb} f_a f_{LU} f_{LS} f_{RS} f_{Lpb} f_{Rpb} \tag{4.3}$$

FIGURE 4.5: Experiment 1 for a simple scenario, upper row shows fitness evolution for 5,25 and 40 generations, lower row shows the MA50 jam length(meters) for Lane 1,Lane 2,Lane 3 and Lane 4 for 5,25 and 40 generations



All the correction parameters f are used to reduce the value of 1900, for example, the number of vehicles turning right,the inclination, the intersection location etc. So I have decided don't use it, keeping the value unchanged, that is to say, a saturation of 1900 veh/h ($N_l = 1$, one lane).

For the second experiment We are using different flow probabilities, in this case I am going to increase the load in two lanes with respect to the others, the pedestrians flow probability is also reduced by a factor of ten. We are going to repeat the previously used number of generations 5,25 and 40 to compare the response of the system, all the results are summarized in 4.7.

FIGURE 4.6: Experiment 2 for a simple scenario, upper row shows fitness evolution for 5,25 and 40 generations, lower row shows the MA50 jam length(meters) for Lane 1,Lane 2,Lane 3 and Lane 4 for 5,25 and 40 generations

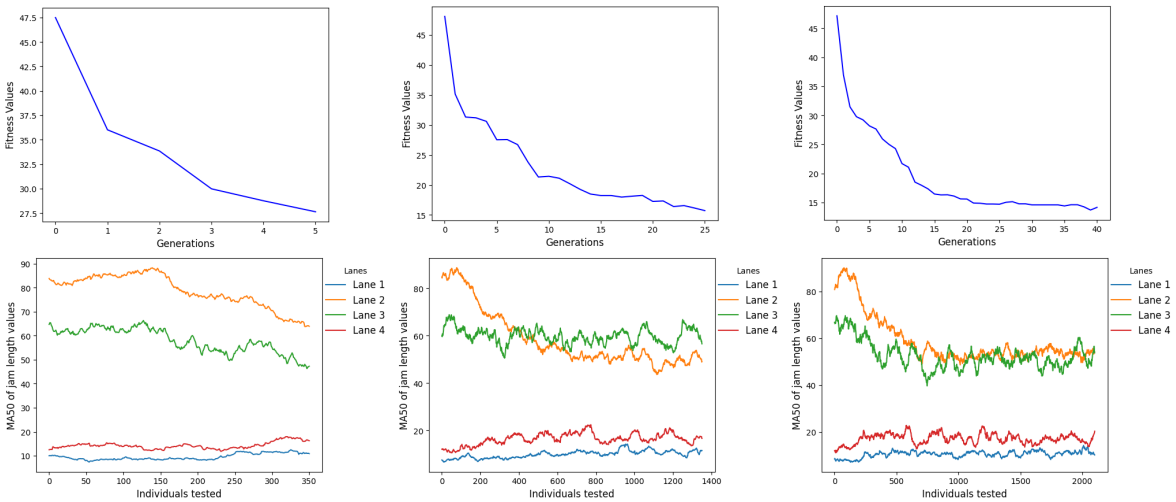


TABLE 4.5: Vehicle flows

Probability	From	To
0.01	North	East
0.01	North	South
0.01	North	West
0.06	West	East
0.06	West	North
0.06	West	South
0.005	East	North
0.025	East	South
0.005	East	West
0.01	South	East
0.02	South	North
0.01	South	West

TABLE 4.6: Pedestrian flows

Probability	Origin	Destiny
0.01	East	West
0.01	North	South
0.01	South	North
0.01	West	East

TABLE 4.7: Experiment 2 in simple scenario

	5 generations	25 generations	40 generations
Duartion	450	450	450
Intensity (veh/h)	[108, 648, 450, 144]	[108, 648, 450, 144]	[108, 648, 450, 144]
Best Time Schedule	[20, 54, 41, 15, 57]	[15, 49, 41, 15, 25]	[18, 57, 28, 17, 31]
Best v2c Time Schedule	[28, 52, 55, 17, 56]	[16, 57, 43, 22, 41]	[16, 60, 50, 21, 53]
v2c value	0.562	0.577	0.586
Execution time	206 sec.	595 sec.	877 sec.

In panel of plots 4.6 is shown the evolution of the second experiment execution.Looking at the data obtained and the plots, We can conclude that in a short number of generations the algorithm is able to obtain quite good solutions in both different flow scenarios. So, employing low execution time, We are able to find an acceptable solutions in terms of fitness values to the optimization problem. We can compare this results with the jam length behavior for all the individuals along the simulation.

In this case, the convergence of all jam length is not possible, because of the upper and lower time bounds, so the behavior is the expected, the closer approaches, in terms of traffic load, converge minimizing its jam length, those approaches with a low load remains quite stable. This simulations demonstrate that the system is suitable for being used in real time scenarios as an adaptive control system.

4.4.2 Results in an evolutionary scenario

With the aim of testing the resilience and the suitability of the algorithm in a closer to real environment, which can experiment abrupt changes along time, I have proposed a script for testing purposes, which implements the discussed algorithm and performs a random change in the flow probabilities along time,besides, as in the simple experiment, the seed for the random number generator is changed for each execution of the algorithm, so each time we run the program we get different random values, and even each time an individual is tested, the sumo simulator will behave different.

Each experiment execution corresponds with the computation of all generations against certain flow values, then, the flow parameters, are changed randomly, being updated accordingly with the time frame 4.5, that corresponds with the time to compute 450 simulation steps or, in other words, the train time t_{train} . The idea is that the population would be checking its performance against the environment regularly, getting in this way, the best suited individuals, plans, in our scope, for the current environment state.

For this test, all the parameters are the same as the used previously, but a reduction in the number of generations, fixed to three in this case. The number of experiments is a new value added in this script and it is set to ten and it states how many times the complete algorithm is executed, other parameter named "bound" is used to control how is randomized the change between the experiments.

The initial population is only computed the first time ,then, the solution for each experiment round evolves from this original set of individuals. As I said before, when a experiment finishes, the probability for a flow to be issued is randomly changed, doing that, I am trying to emulate the changes that the traffic might experiment in a real environment. After an experiment is performed the plans stored in the HOF are tested against the test environment, then, once actualized its fitness according to its performance, the non dominated plan is selected between them, this one, should be finally the selected plan to apply to the real scenario.

The initial probabilities are the same as in the first experiment and can be seen in 4.2 and 4.3. This values are changed adding or subtracting a value uniform at random selected, in the range $[-0.02, 0.02]$, this adjustment, apparently slight, is actually quite abrupt in the support function boundaries,because, if we compute the expected value for this binomial process, in the case of 0.02, for an hour, i.e. 3600 seconds, as we see in 4.4, would suppose increase the expected volume traffic in 72 vehicles/hour, in a short time.

TABLE 4.8: Data collected in experiment 1 against test environment (Gt=Green Time, Os=Offset, Std=Standard deviation, SoG=Sum of gradients)

Experiment ($\triangle t$)	Intensity (veh/h/lane)	Gt (sec/stage),Os (sec)	Mean	Std	SoG
1	[432, 108, 180, 324]	[60, 15, 27, 49, 42]	14.0	5.0	4.0
2	[347, 273, 219, 360]	[53, 16, 28, 34, 17]	14.0	8.0	6.0
3	[298, 362, 278, 305]	[34, 26, 24, 53, 40]	14.0	5.0	10.0
4	[452, 517, 243, 177]	[40, 16, 49, 59, 59]	15.0	13.0	20.0
5	[427, 628, 162, 204]	[53, 16, 28, 26, 40]	18.0	7.0	12.0
6	[397, 681, 70, 297]	[34, 31, 19, 59, 59]	20.0	6.0	14.0
7	[405, 578, 135, 381]	[34, 27, 15, 28, 23]	22.0	15.0	19.0
8	[340, 699, 106, 349]	[35, 47, 19, 34, 34]	28.0	10.0	39.0
9	[433, 688, 156, 357]	[34, 47, 19, 34, 34]	29.0	19.0	57.0
10	[417, 828, 260, 377]	[34, 47, 19, 34, 34]	26.0	22.0	32.0

TABLE 4.9: Data collected in experiment 1 against test environment (Gt=Green Time, Os=Offset, Std=Standard deviation, SoG=Sum of gradients)

Experiment ($\triangle t$)	Intensity (veh/h/lane)	Gt (sec/stage), Os (sec)	Mean	Std	SoG
1	[432, 108, 180, 324]	[57, 26, 24, 53, 26]	21.0	10.0	11.0
2	[347, 273, 219, 360]	[53, 16, 15, 45, 32]	33.0	11.0	24.0
3	[298, 362, 278, 305]	[37, 27, 15, 23, 18]	32.0	17.0	22.0
4	[452, 517, 243, 177]	[42, 27, 19, 34, 23]	52.0	19.0	54.0
5	[427, 628, 162, 204]	[53, 16, 28, 26, 40]	43.0	40.0	31.0
6	[397, 681, 70, 297]	[34, 31, 19, 28, 23]	39.0	30.0	2.0
7	[405, 578, 135, 381]	[34, 27, 15, 28, 23]	37.0	32.0	63.0
8	[340, 699, 106, 349]	[28, 47, 15, 23, 95]	54.0	29.0	51.0
9	[433, 688, 156, 357]	[35, 47, 19, 34, 74]	67.0	19.0	15.0
10	[417, 828, 260, 377]	[34, 47, 19, 34, 34]	66.0	18.0	19.0

In the tables 4.8, 4.12 and 4.10 are shown the result yielded by the execution of ten simulation rounds, the first table shows the traffic intensity for this interval, the best plan obtained tested against the reference environment and its fitness values, the same apply for the second table, but in this case is shown the individual into the HOF with the best performance tested in the test environment, finally, the third table shows the volume to capacity ratio of the former individuals for each round.

$$E[X] = np = 3600 \cdot 0.02 = 72 \tag{4.4}$$

$$t_i = t_{i-1} + \triangle t = t_{i-1} + 450\ steps \equiv t_{i-1} + t_{train} \tag{4.5}$$

In the plots 4.7 and 4.8 we can see the evolution of time allocation for each stage along with number of vehicles evolution. The fitness valus are also depicted, this give us an idea of its behavior from round to round.

Looking at the retrieved values, we see that, the green time values assigned to

TABLE 4.10: Volume to Capacity ratio in experiment 1 (Gt=Green time, Os=Offset)

Experiment ($\triangle t$)	Gt(sec/stage), Os (sec)	Volume to Capacity Ratio
1	[57, 26, 24, 53, 26]	0.61
2	[47, 33, 19, 34, 57]	0.65
3	[42, 27, 32, 39, 52]	0.74
4	[46, 38, 32, 32, 25]	0.77
5	[35, 47, 27, 28, 34]	0.83
6	[46, 51, 26, 28, 34]	0.80
7	[35, 47, 19, 34, 34]	0.80
8	[42, 47, 19, 34, 34]	0.88
9	[35, 47, 19, 34, 74]	0.86
10	[35, 47, 19, 34, 34]	0.96

each lane are quite correlated with the expected value of vehicles/hour, I mean, the green time increases with the traffic load in this lane and decreases conversely. At this point, it is interesting to mention that this correlation might not be a very good sign in many situations, in more complex scenarios, may exist traffic recirculation, for example, people trying to find a parking place close to a social event. In this cases, might be interesting reduce the time for a future better performance . This is other advantage of using GA, which should be resilient in this situations, furthermore, in my case , the optimization function not only grant a reduction in the jam length, it is also designed to spread the cars along the intersection approaches as balanced as possible, this is the motivation of using the three objective functions, so this correlation, is not as relevant as might seem.

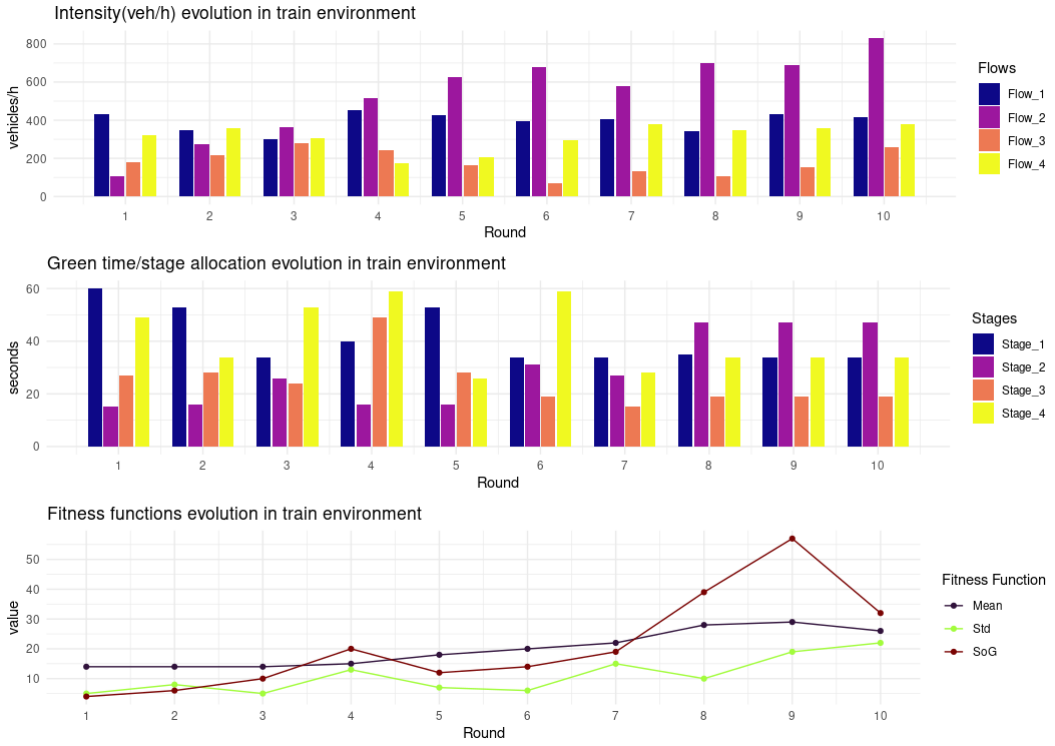
Although, the number of generations is only three,for each experiment round, the algorithm behaves quite well, even in a highly volatile environment, as this case, where the changes are quite abrupt .We can note the effect taking a glance to the fitness peaks in 4.9.

The HOF is updated for each experiment with its performance against the test environment, this adds diversity to the population because must perform well in two scenarios with random at uniform behaviors. If its performance is good enough for the current conditions, some individuals, may survive into the HOF set between experiment rounds.

At the light of the results obtained in the long term with only three generations per experiment, might be interesting increase the number of generations to compare with the current parameter,so I have doubled the number of generations per experiment. The results are depicted in 4.10 .The plot shows that, in the long term, the algorithm seems able to recover from the bad performance during the initial experiments, although, this will also depend the way in that the probability evolves along the experiment. Besides, the data suggest that, the fitness parameter behaves better in the short term, using more generations.

In the next plot 4.11 we see the output obtained launching the same experiment, setting the parameters as the former,but in this case, repeating 20 times, instead of 10 as previously did. This run of the process illustrates the difficult that can be keeping a stable performance in a scenario as random as this one. At the beginning a optimal solution with very low fitness values was reached very quickly, but then, a correction had to be carried out. Despite, the population set was able to correct the performance

FIGURE 4.7: Plot from data collected in experiment 1 against train environment (Plot 1 shows expected veh/h/lane, Plot2 shows green time allocated/stage, Plot 3 shows the fitness functions evolution)



spikes, keeping very good fitness parameters along the training process.

The intermediate experiment execution time is very low,as low as roughly 75 seconds,with the resources listed in A. With this execution time find a solution can become very quick .In the Linux machine used,I am obtained even lower execution time applying a re-nice to the simulator process, giving in this way greater priority execution.

Since, It would be difficult experiment such an exceptional change in the expected intensity traffic in only 75 seconds (unless and exceptional event happens),is interesting repeat the experiment using a narrower random at uniform function support, in detail, $[-0.005, 0.005]$, this suggests an expected intensity traffic change of ± 18 vehicles/hour per iteration, at boundary values. This situation, in my opinion, is closer to real traffic conditions of volatility in most of situations. I have retrieved the results depicted in the tables 4.11, 4.12 and 4.13.

As in the previous experiment, for visualization purposes, two panel plots have been generated, in 4.12 and 4.13 we can see a more soft variation with respect to the previous experiment, and closer to zero fitness values.

Here the algorithm is ,obviously, much more stable. It is able to adapt very quickly to this slighter traffic flow variations, and performs better in the long term, being able to remain stable in quite low fitness values. Other interesting observation comes from the fact that, the offset parameter, i.e the fifth element that every chromosome has, also seems to evolve taking in general low values, this might be explained by the number of steps selected for the simulation, high offset values will probably worsen the jam parameters.

In real traffic management infrastructures, many times, is mandatory by design,

TABLE 4.11: Data collected in experiment 2 against test environment (Gt=Green Time, Os=Offset, Std=Standard deviation, SoG=Sum of gradients)

Experiment ($\triangle t$)	**Intensity (veh/h/lane)**	**Gt (sec/stage), Os (sec)**	**Mean**	**Std**	**SoG**
1	[432, 108, 180, 324]	[35, 20, 20, 24, 33]	13.0	6.0	3.0
2	[390, 104, 159, 341]	[43, 15, 19, 26, 56]	13.0	7.0	0.0
3	[377, 85, 176, 331]	[51, 18, 18, 27, 25]	12.0	3.0	2.0
4	[361, 95, 153, 294]	[51, 18, 18, 27, 34]	9.0	7.0	7.0
5	[344, 125, 174, 268]	[51, 18, 18, 27, 34]	9.0	6.0	14.0
6	[344, 121, 146, 285]	[36, 21, 18, 27, 25]	9.0	4.0	6.0
7	[337, 109, 137, 284]	[43, 17, 26, 42, 33]	9.0	2.0	3.0
8	[338, 115, 143, 304]	[45, 21, 18, 27, 25]	7.0	3.0	2.0
9	[338, 97, 163, 329]	[45, 15, 26, 27, 33]	9.0	3.0	11.0
10	[327, 104, 188, 297]	[43, 21, 18, 27, 33]	9.0	3.0	12.0

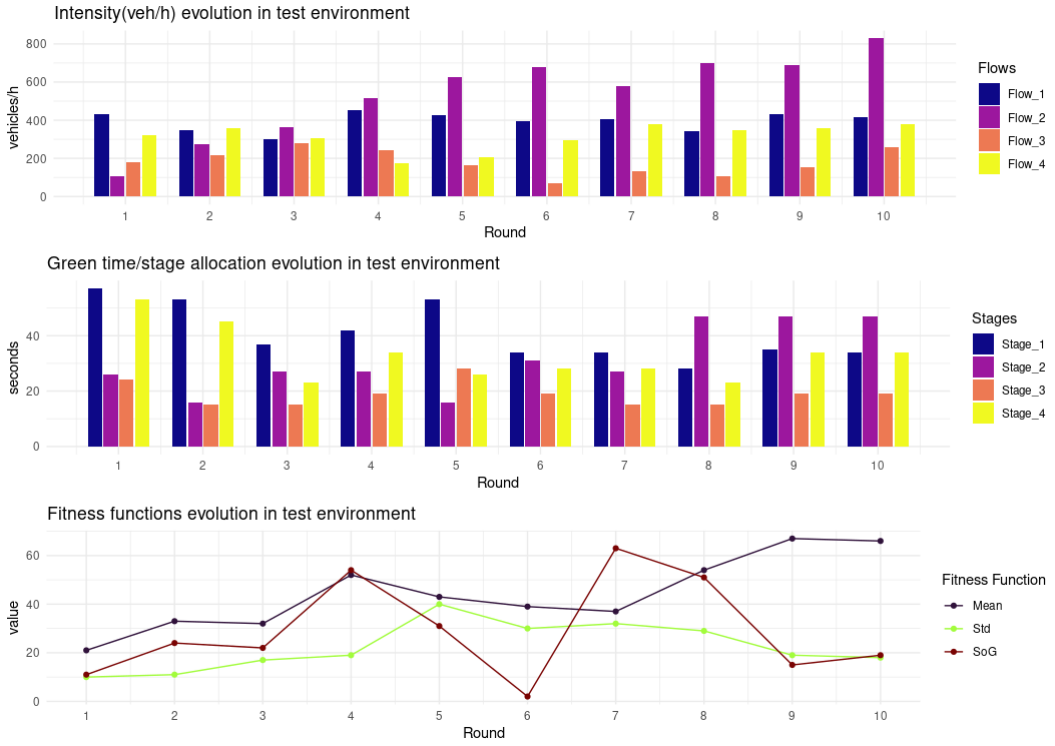
TABLE 4.12: Data collected in experiment 2 against test environment (Gt=Green Time, Os=Offset, Std=Standard deviation, SoG=Sum of gradients)

Experiment ($\triangle t$)	**Intensity(veh/h/lane)**	**Gt(sec/stage), Os(sec)**	**Mean**	**Std**	**SoG**
1	[432, 108, 180, 324]	[46, 28, 20, 42, 56]	16.0	10.0	9.0
2	[390, 104, 159, 341]	[43, 15, 19, 26, 56]	30.0	26.0	80.0
3	[377, 85, 176, 331]	[51, 18, 18, 27, 25]	19.0	10.0	5.0
4	[361, 95, 153, 294]	[45, 15, 18, 27, 25]	17.0	5.0	12.0
5	[344, 125, 174, 268]	[35, 18, 26, 42, 33]	28.0	18.0	38.0
6	[344, 121, 146, 285]	[39, 18, 36, 38, 7]	18.0	11.0	15.0
7	[337, 109, 137, 284]	[36, 18, 18, 27, 33]	15.0	6.0	9.0
8	[338, 115, 143, 304]	[45, 21, 18, 27, 25]	19.0	10.0	5.0
9	[338, 97, 163, 329]	[45, 15, 26, 27, 33]	15.0	11.0	43.0
10	[327, 104, 188, 297]	[43, 21, 18, 27, 33]	13.0	7.0	5.0

TABLE 4.13: Volume to Capacity ratio in experiment 2 (Gt=Green time, Os=Offset)

Experiment ($\triangle t$)	**Gt (sec/stage**, Os (sec)	**Volume to Capacity Ratio**
1	[53, 29, 37, 34, 15]	0.61
2	[46, 18, 26, 58, 33]	0.61
3	[45, 18, 19, 58, 25]	0.62
4	[51, 18, 20, 42, 33]	0.59
5	[45, 18, 26, 42, 33]	0.53
6	[39, 18, 36, 38, 7]	0.56
7	[43, 17, 26, 42, 33]	0.55
8	[43, 21, 18, 27, 42]	0.54
9	[41, 18, 26, 42, 33]	0.54
10	[35, 18, 18, 27, 33]	0.58

FIGURE 4.8: Plot from data collected in experiment 1 against test environment(Plot 1 shows expected veh/h/lane, Plot2 shows green time allocated/stage, Plot 3 shows the fitness functions evolution)



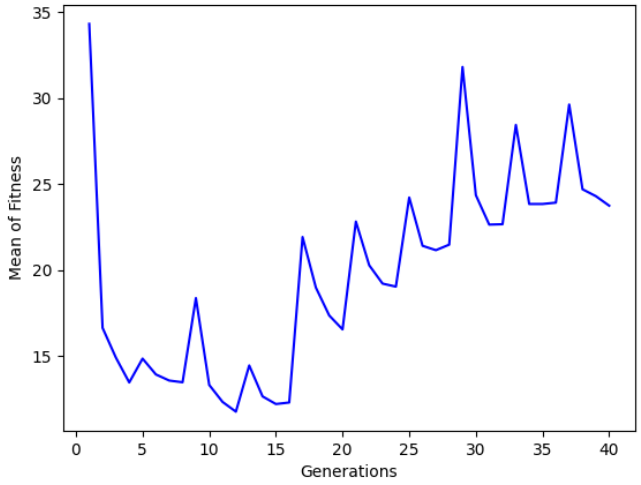
keep a fixed cycle value for a group of TL intersections. This is used to keep the different TL synchronized by means of the offset parameter, and in this way, allowing a car that leaves an intersection find the next light in green state. This approach, is useful in streets with a high traffic load and we want the flows moving across, enter and leaving the route in as little steps as possible.

To measure the performance of the algorithm with this constraint, the test application includes a function which adapt the cycle time of the individuals of our population, returning its parameters adapted to this cycle, the function works removing or adding, either the remaining or necessary time, using an approach that stem from a mix between a random and a fairy distribution, the result of the extra time is divided by the length of the plan an fairly added or subtracted, the remainder is assigned randomly, note that, the lower and upper bounds must be respected because are limited by design, so, the remainder is used to tune the division in order to respect this constraint.

In this scenario it is valuable see how the jam evolves during the execution along with the fitness parameters, in the plots 4.14 we see the fitness evolution and the jam behavior for the four approaches with the current probability parameters and the adaptation cycle turned on for a cycle of 120 seconds, note that, the adaptation is only performed with the individuals into the HOF set, because they would be those, hypothetically used, to be applied in a real scenario.

In 4.15 I have repeated the study for the highly randomized scenario, keeping the rest of parameters fixed, but the number of generation per experiment which is changed to six. At the light of the graph, the results seems bad, because the jam length increases for some lanes,but if we delve into the data, we can see that the simulation begins with a flow of $F(0) = [432, 108, 180, 324]$ and finishes with a flow of

FIGURE 4.9: Mean fitness evolution [Ngen=3, Nexp=10, ProbChange=[-0.02,0.02]]



$F(10) = [174, 255, 250, 585]$, the best scored green time allocation for this experiment in the first round was [60, 23, 24, 38, 31] and for the last round [15, 23, 27, 49, 52], the green times are correlated with the flow parameters and evolve accordingly.

With this information in mind, actually, the results are not so bad, the fitness performance depends, for example,on the mean of the jam length, and this parameter is tied to constrains imposed by the environment and the intersection characteristics, this fact limits the lower values that the fitness functions can take, we only can increase the green time until a limit, so, the objective of the algorithm should be find the best of the allowed solutions into this limited space of solutions, which vary with time,so, may be broader or narrower that in the previous round of the experiment. If we take a glance at the mean fitness evolution plot **??** we see that the fitness parameter is corrected very quickly,drawing a saw-tooth wave, modulated by the flow change, this demonstrate resilience capacity.

In other try, depicted in 4.16, the random values tend toward lower values of traffic volume, then, in this case, the fitness values can be greatly improved.

FIGURE 4.10: Mean fitness evolution [Ngen=6, Nexp=10, ProbChange=[-0.02,0.02]]

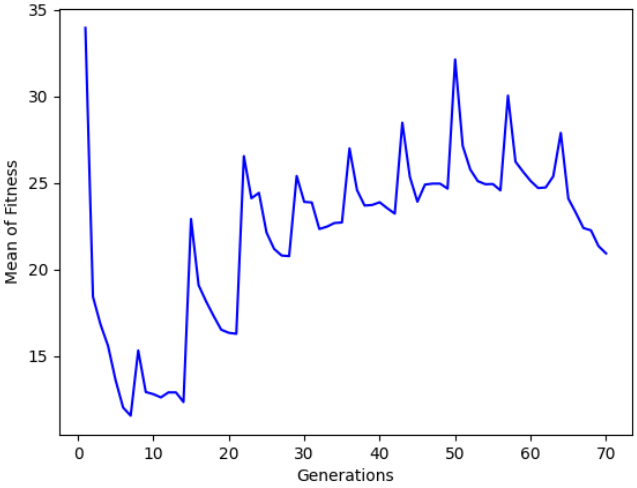


FIGURE 4.11: Mean fitness evolution [Ngen=6, Nexp=20, ProbChange=[-0.02,0.02]]

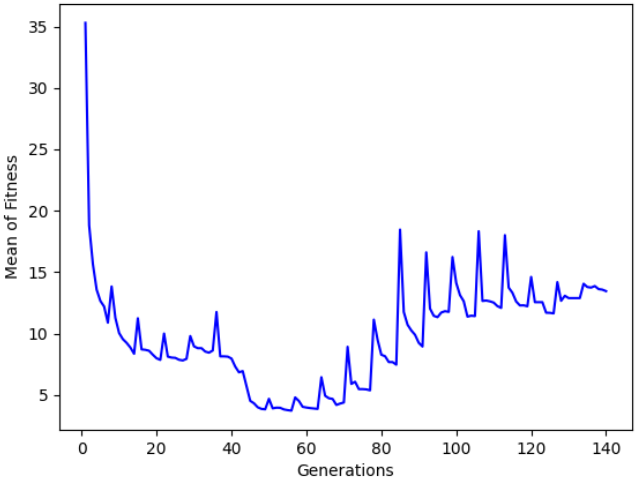


FIGURE 4.12: Plot from data collected in experiment 2 against train environment (Plot 1 shows expected veh/h/lane, Plot2 shows green time allocated/stage, Plot 3 shows the fitness functions evolution)

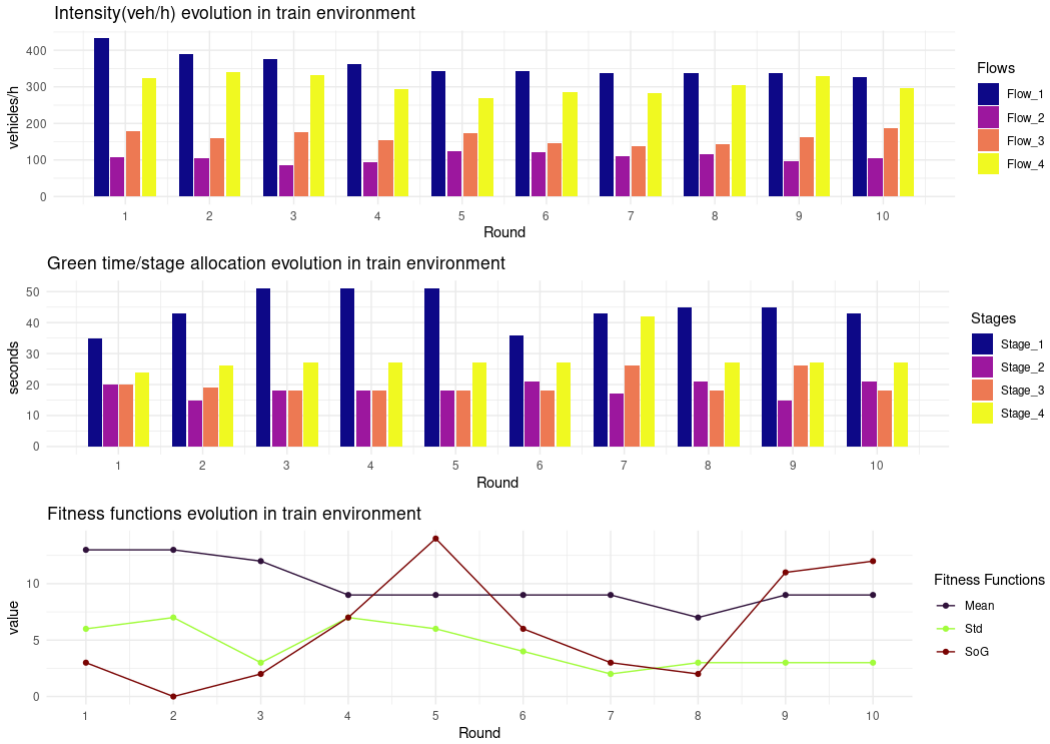


FIGURE 4.13: Plot from data collected in experiment 2 against test environment (Plot 1 shows expected veh/h/lane, Plot2 shows green time allocated/stage, Plot 3 shows the fitness functions evolution)

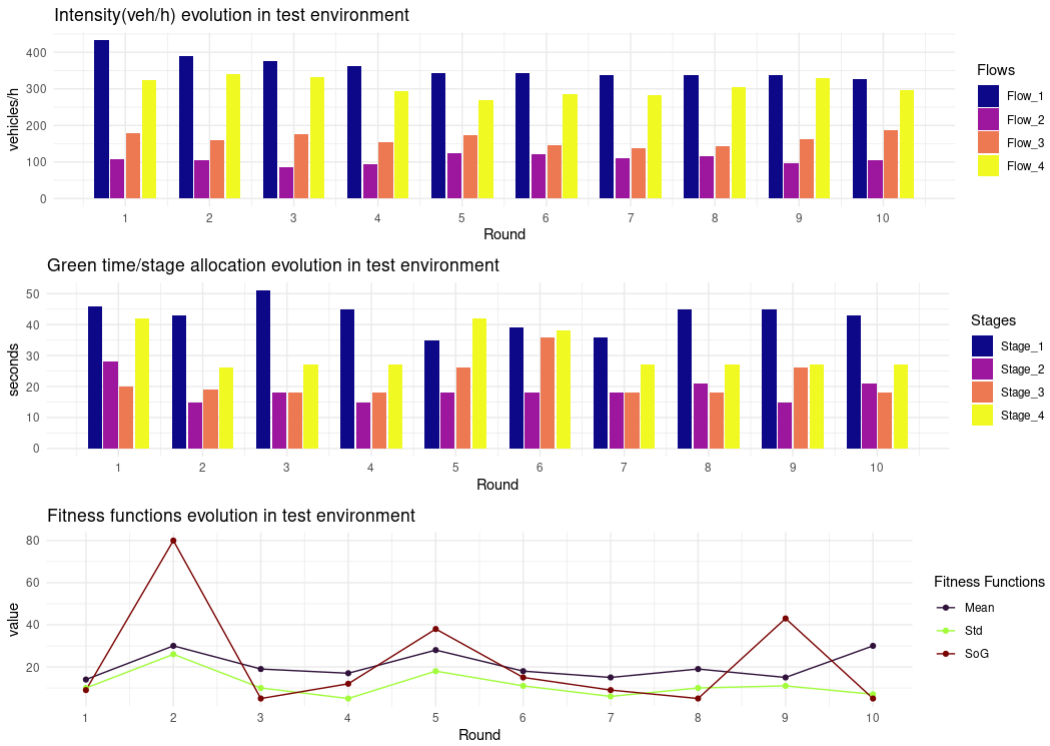


FIGURE 4.14: Fitness evolution and MA50 Jam length with cycle adapted [Ngen=3, Nexp=10, ProbChange=[-0.005,0.005], cycle=120]

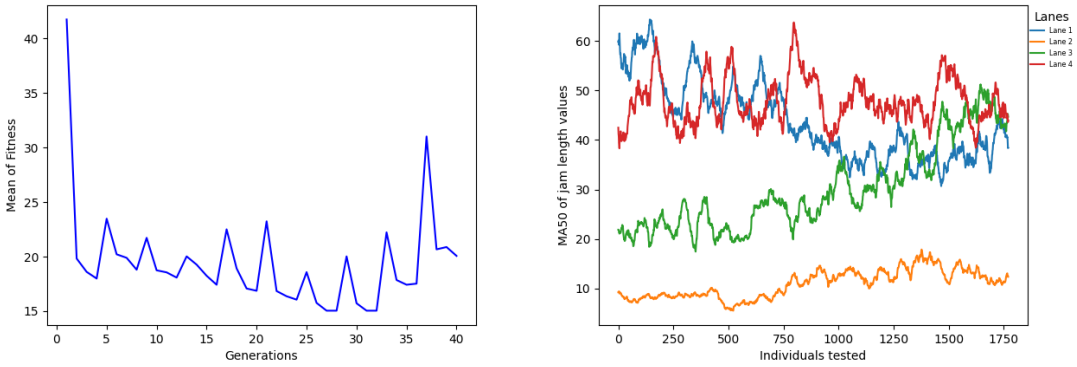


FIGURE 4.15: Fitness evolution and MA50 Jam length with cycle adapted [Ngen=6, Nexp=10, ProbChange=[-0.02,0.02], cycle=120]

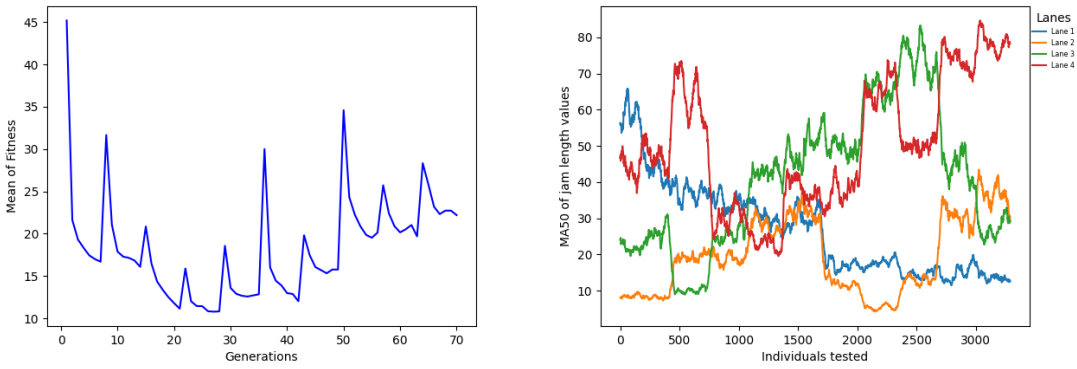
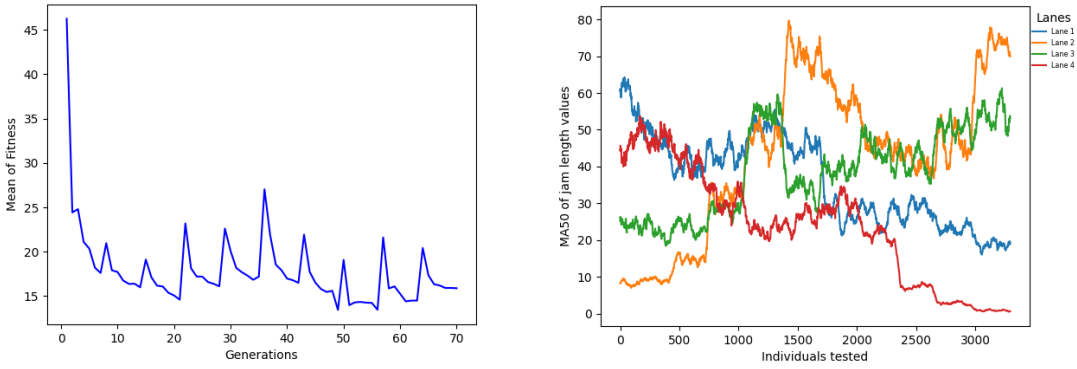


FIGURE 4.16: Fitness evolution and MA50 Jam length with cycle adapted [Ngen=6, Nexp=10, ProbChange=[-0.02,0.02], cycle=120]



Chapter 5

Conclusions and future work

5.1 Conclusions

The algorithm has shown during the experiments carried out quite good performance .The execution of a single training process is able to reach optimal fitness scores very quickly. On the other hand, launching the algorithm in a evolutionary scenario has demonstrate its resilience using a short number of generations, even in highly randomized scenarios. This behavior made it a suitable adaptive tool for real time scenarios, due to its capability to recover from environment changes, even when this changes are large, besides, the experiments have shown that, the more time the agent is "on-line" the better its behavior tends to be.

5.2 Future research

Obviously, the algorithm is limited to achieve its objectives by physical conditions which fall out of its scope, for example, the number of lanes per approach, the type of vehicles crossing the intersection, the type of flows and its timing, the traffic volume, between others. All this parameters are out of our control when only an agent is present in the net, otherwise, the volume traffic, for example, might be indirectly controlled if we implement the agent in all the intersections of the same area. In this case, the net where the agent is running should be able to adapt the traffic using a decentralized approach,where each agent is interacting, without being aware of the existence one of each other. We couldn't change the number of lanes, but we might control the way in that the flows are issued.

This is the reason why, in my opinion,would be valuable testing the algorithm capabilities in a scenario where several decentralized agents interact in an oblivious way.

Study if the system is actually able to reach certain stability level would be an interesting challenge. The automated parameter selection would be other feature to be added, this would accelerate our possibilities to test the performance in a broader range of different situations adding value to the solution.

The automated parameter selection might be carried out using a parallel environment which simulates the current, this one, might be fully independent from the former, working in a parallel process or even previous to the implementation using former real data. This environment would be in charge of the exploration of the parameters with better performance in different traffic situations for the environment. Either a supervised or an unsupervised learning approach, based on neural networks might be a suitable solution, even a GAN algorithm where two networks, one issuing flows trying to worsen the fitness scores and other changing the hyperparameters trying to improve the fitness outcome, could be fine.

Furthermore, we must be aware that, not always, we have at our disposal all the desired hardware capabilities to perform the computations, besides, a reduction in the execution time will led us to a reduction in training time per generation, allowing us to increase the population length, breeding longer offspring and raise the number of generations that we are able to execute in the time slot at our disposal between queries, issuing better fitness performances. These are the reasons why it would be very interesting find the way to execute the quickest algorithm possible. In our case, the major limitation comes from the simulator, since we are using an external solution, which might be difficult to adapt to our requirements, It would be great explore the possibility of design a simulator ad-hoc able to be executed in a GPU, which should improve the execution times. We do not need, actually, a very complex simulator, we are using a very little part of the sumo capabilities, so, this is also an interesting field of study.

Appendix A

Resources

A.1 Experimental Equipment

The experiments have been performed in a laptop with the following software and hardware characteristics:

- **SO**: Ubuntu 20.04.2 LTS
- **Linux Kernel**: 5.4.0-77-generic
- **Processor**: Intel® Core™ i7-6500U CPU @ 2.50GHz × 4 64 bits
- **RAM Memory**: 15,6 GiB
- **Graphics Card**: Intel Corporation Skylake GT2
- **Programming Language**: Python 3.8.10
- **IDE**: PyCharm 2021.1.3 (Community Edition)

A.2 Simulator

Eclipse SUMO Version 1.4.0
Build features: $x8664 - pc - linux - gnuGDALGUI$
Copyright (C) 2001-2019 German Aerospace Center (DLR) and others; https://sumo.dlr.de
Eclipse SUMO Version 1.4.0 is part of SUMO.
http://www.eclipse.org/legal/epl-v20.html
SPDX-License-Identifier: EPL-2.0

A.3 Python Frameworks

The python frameworks used are:

- Name: deap
  Version: 1.3.1
  º Summary: Distributed Evolutionary Algorithms in Python
  Home-page: https://www.github.com/deap
  Author: deap Development Team
  Author-email: deap-users@googlegroups.com
  License: LGPL

- Name: tensorflow
  Version: 2.4.1
  Summary: TensorFlow is an open source machine learning framework for everyone.
  Home-page: https://www.tensorflow.org/
  Author: Google Inc.
  Author-email: packages@tensorflow.org
  License: Apache 2.0

- Name: tf-agents
  Version: 0.7.1
  Summary: TF-Agents: A Reinforcement Learning Library for TensorFlow
  Home-page: https://github.com/tensorflow/agents
  Author: Google LLC
  Author-email: no-reply@google.com
  License: Apache 2.0

- Name: sumolib
  Version: 1.8.0
  Summary: Python helper modules to read networks, parse output data and do other useful stuff related to the traffic simulation SUMO
  Home-page: https://sumo.dlr.de/docs/Tools/Sumolib.html
  Author: DLR and contributors
  Author-email: sumo@dlr.de
  License: EPL-2.0

- Name: traci
  Version: 1.8.0
  Summary: The pure python version of the TraCI API to communicate with the traffic simulation SUMO
  Home-page: https://sumo.dlr.de/docs/TraCI/InterfacingTraCIfromPython.html
  Author: DLR and contributors
  Author-email: sumo@dlr.de
  License: EPL-2.0

Appendix B

Python Code

B.1 Repositories

The code is served in two repositories, github and docker. The code was designed to work in Linux machines, and has several dependencies that could be difficult to accomplish, this fact was the motivation to implement a docker image built on the top of a docker ubuntu image, with all the dependencies satisfied. The instructions to run the software are in the github repository , I am sharing here the different links to access the repositories:

Github Repository Project
Docker Project Repository
Registry to download the docker image: suarna/gatraffic:latest

B.2 Files

Here i give a breaf explanaiton of the python files and its function.

B.2.1 gatraffic.py

This file is one of the main script functions of the genetic algorithm traffic implementation, so we can consider it the core of the proposed solution. The simulation parameters are also defined here. This script carry out a single execution of the algorithm according to the configured. parameters.

B.2.2 tlenvironment.py

The tensor flow environment is created here. The class contained in this file is used to interact with sumoconnector interface,and in this way,controlling the init,finish and stepping of simulation and the retrieval of information from it. Each step returns a reward, which is always zero until the last step, where the jam length value for each lane is returned as the final reward. This solution stems from reinforcement learning approaches.

B.2.3 paramstorage.py

Into this file several static methods are defined. They are tools used to manipulate the content of xml config files used by the sumo simulator.

B.2.4 test-gatraffic.py

This file is quite similar to gatraffic.py. In this case, instead of performing a single execution of the algorithm, the process is repeated several rounds. The aim of this approach is emulate real conditions where the environment changes from time to time, and in this way, study the resiliency of the solution when the traffic vehicle flows are evolving randomly over time,below is the code of this script.

```
import random as rd
from datetime import datetime
import matplotlib.pyplot as plot
import copy

import numpy as np
import traci
import csv
from deap import algorithms
from deap import base
from deap import creator
from deap import tools

import gatraffictoolbox
import paramstorage
import sumoconnector
import trafficinteract

initial_time = datetime.now()
init_state, n_phases, np_phases, n_steps, det_ids_list, offset =
    trafficinteract.getinfo(
    "Nets/SimpleNet/tls.xml",
    "Nets/SimpleNet/net.net.xml",
    "Nets/SimpleNet/additional.add.xml",
    "Nets/SimpleNet/testdemandpedestrian.rou.xml")

# LOAD HYPER-PARAMETERS FROM FILE
hyper_params = params('@param_test.txt')
print("Loaded parameters are: {}".format(hyper_params))
print(hyper_params)
for key, val in hyper_params.items():
    exec(key + '=val')

# DECLARE MORE PARAMETERS
CHROMOSOME_LENGTH = np_phases + 1
MAX_PH_TIME = CYCLE - (MIN_PH_TIME * np_phases)

# Define seed
rd.seed(RANDOM_SEED)

intensity =
    paramstorage.get_flow("Nets/SimpleNet/testdemandpedestrian.rou.xml")
# Store a tmp file of demand data file
xml_copy =
    paramstorage.temp_xml("Nets/SimpleNet/testdemandpedestrian.rou.xml")
print("\033[93mThe flow in vehicles/h is: {}\033[0m".format(intensity))

if SINGLE:
```

```
    weights = (-1.0,)
    creator.create("FitnessSingle", base.Fitness, weights=weights)
    creator.create("Individual", list, fitness=creator.FitnessSingle)
else:
    # (divergence, mean, std)
    weights = (-1.0, -1.0, -1.0)
    # Define creator
    # Define minimize strategy
    creator.create("FitnessMulti", base.Fitness, weights=weights)
    # Gen class
    creator.create("Individual", list, fitness=creator.FitnessMulti)


#
    ----------------------------------------------------------------------------------------

# Define score function
def score(individual, env):
    return sim.get_score(env, individual)


# Define hof similarity function to limit the hof length if genotype is the
    same
def pareto_eq(ind1, ind2):
    return np.any(ind1.fitness.values == ind2.fitness.values)


#
    ----------------------------------------------------------------------------------------------------


# Define toolbox & register register operators
toolbox = base.Toolbox()
toolbox.register("Action", rd.randint, MIN_PH_TIME, MAX_PH_TIME)
toolbox.register("individualCreator", tools.initRepeat, creator.Individual,
    toolbox.Action, CHROMOSOME_LENGTH)
toolbox.register("populationCreator", tools.initRepeat, list,
    toolbox.individualCreator)

# Define statistics
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("mean", np.mean)
stats.register("max", np.max)
stats.register("min", np.min)
stats.register("std", np.std)

if SINGLE:
    hof = tools.HallOfFame(maxsize=HOF_SIZE)
else:
    # hof = tools.ParetoFront(similar=pareto_eq)
    hof = tools.ParetoFront(similar=pareto_eq)

# Run simulation
fit_list = list()
sim = trafficinteract.TrafficEnv(CONFIG_FILE_ROUTE, N_STEPS,
    CHROMOSOME_LENGTH, det_ids_list, SINGLE, fit_list)
sim.runs()
```

```
# Running original environment
ref_env = sim.rune("Reference")
# Running test environment
test_env = sim.rune("Test")

# Upper and lower sequences with limits for genes generation
lower_bound = []
upper_bound = []
# Boundaries for phases
for n in range(0, np_phases):
    lower_bound.append(MIN_PH_TIME)
    upper_bound.append(MAX_PH_TIME)
# Adding the offset limits
lower_bound.append(MIN_OFFS_TIME)
upper_bound.append(MAX_OFFS_TIME)

toolbox.register("evaluate", score, ref_env)

if SEL_AL1:
    toolbox.register("select", tools.selSPEA2)
elif SEL_AL2:
    ref_points = tools.uniform_reference_points(2, np_phases + 1)
    toolbox.register("select", tools.selNSGA3, ref_points=ref_points)
elif SINGLE:
    toolbox.register("select", tools.selBest)
else:
    toolbox.register("select", tools.selNSGA2)
toolbox.register("mate", tools.cxOnePoint)
toolbox.register("mutate", tools.mutUniformInt, low=lower_bound,
    up=upper_bound, indpb=1.0 / CHROMOSOME_LENGTH)

# Creating population
population = toolbox.populationCreator(n=POPULATION_SIZE)

v2c_list = []
lb_list = []
best_list = []
best_test_list = []
intensity_list = [intensity]


def main():
    try:
        for i in range(0, N_EXPERIMENTS):
            print("-------------------------------------------------------------------------------
            print("EXPERIMENT NUMBER {} INITIATED".format(i + 1))
            print("-------------------------------------------------------------------------------
            if i != 0:
                # Update HOF fitness for new environment conditions
                for _ in range(len(hof)):
                    result = sim.get_score(hof.items[_], ref_env)
                    hof.items[_].fitness.values = result
                    if hof.items[_] in pop:
                        pop[pop.index(hof.items[_])].fitness.values = result
            pop, lb = algorithms.eaMuPlusLambda(population,
                                                toolbox,
                                                MU,
```

```
                                        LAMBDA,
                                        P_CROSSOVER,
                                        P_MUTATION,
                                        MAX_GENERATIONS,
                                        stats,
                                        hof,
                                        True)

            lb_list.append(lb)
            # Get a copy of the best hof item
            best = copy.deepcopy(hof.items[0])
            best_list.append(best)
            print("\n\033[0;31;40mHOF content is: ", hof)
            print("Best Plan = {}\033[0m".format(best))

            # Apply plan to test env and read result
            test_result = []
            for _ in range(len(hof)):
                if ADAPT:
                    # Adapt to cycle
                    plan = gatraffictoolbox.adapt(hof.items[_], CYCLE,
                        MIN_PH_TIME, MAX_PH_TIME)
                    result = sim.get_score(plan, test_env)
                    # Update fitness value of hof item
                    hof.items[_].fitness.values = result
                    test_result.append(hof.items[_])
                    print("The test score for individual {} adapted from {}
                        is : {}".format(plan, hof.items[_], result))

                else:
                    result = sim.get_score(hof.items[_], test_env)
                    # Update fitness value of hof item
                    hof.items[_].fitness.values = result
                    test_result.append(hof.items[_])
                    print("The test fitness score for individual {} is :
                        {}".format(hof.items[_], result))

            # Get volume to capacity values
            v2c = []
            new_intensity =
                paramstorage.get_flow("Nets/SimpleNet/testdemandpedestrian.rou.xml")
            for _ in range(len(hof)):
                if i == 0:
                    v2c.append(np.mean(sim.v2c(hof.items[_], intensity)))
                else:
                    v2c.append(np.mean(sim.v2c(hof.items[_], new_intensity)))
            min_idx = np.argmin(v2c)
            v2c_list.append([hof.items[min_idx], v2c[min_idx]])

            # Select non dominated from the test set
            best_test = tools.selNSGA2(test_result, 1)
            best_test_list.append(hof.items[hof.items.index(best_test[0])])

            # Set best test offset value in the traffic light logic
            paramstorage.set_offset("Nets/SimpleNet/tls.xml",
                best_test[len(best_test) - 1])
```

```
        # Tuning prob flow adding or subtracting a (random(-bound,bound))
        paramstorage.set_flow("Nets/SimpleNet/testdemandpedestrian.rou.xml",
            BOUND)
        if i != 0:
            intensity_list.append(paramstorage.get_flow("Nets/SimpleNet/testdemandpedestria
        print("\033[93mThe new flow is:
            {}\033[0m".format(paramstorage.get_flow(
            "Nets/SimpleNet/testdemandpedestrian.rou.xml")))

        print("-------------------------------------------------------------------------------
        print("EXPERIMENT FINISHED")
        print("-------------------------------------------------------------------------------

    # Close environments
    traci.switch("Test")
    sumoconnector.close()
    traci.switch("Reference")
    sumoconnector.close()
    traci.switch("default")
    sumoconnector.close()

    print("Evolution:")
    for lb in lb_list:
        print(lb)
    lb_dict = lb_list[0][0]
    keys = lb_dict.keys()
    lb_url = 'data/logbook_' +
        datetime.now().strftime('%m_%d_%Y-%H:%M:%S') + '.csv'
    with open(lb_url, 'w', newline='') as f:
        dw = csv.DictWriter(f, keys)
        dw.writeheader()
        for l in lb_list:
            dw.writerows(l)

    best_csv = []
    print("Best individual per experiment:")
    idx = 0
    for best in best_list:
        print("Experiment: {} # Best: {} # Fitness value: {}".
              format(idx+1, best, best.fitness.values))
        best_csv.append([idx+1, intensity_list[idx], best,
            best.fitness.values])
        idx += 1
    with open('data/best_' +
        datetime.now().strftime('%m_%d_%Y-%H:%M:%S') + '.csv', 'w',
        newline='') as f:
        writer = csv.writer(f)
        writer.writerow(["Experiment", "Intensity", "Best Plan", "Best
            Fitness"])
        writer.writerows(best_csv)

    best_test_csv = []
    print("Best test individual per experiment:")
    idx = 0
    for best_test in best_test_list:
        print("Experiment: {} # Best test: {} # Fitness test value: {}".
              format(idx+1, best_test, best_test.fitness.values))
```

```
            best_test_csv.append([idx+1, intensity_list[idx], best_test,
                best_test.fitness.values])
            idx += 1
        with open('data/best_test_' +
            datetime.now().strftime('%m_%d_%Y-%H:%M:%S') + '.csv', 'w',
            newline='') as f:
            writer = csv.writer(f)
            writer.writerow(["Experiment", "Intensity", "Best Test Plan",
                "Best Fitness"])
            writer.writerows(best_test_csv)

        v2c_csv = []
        print("Best mean value of volume two capacity ratio:")
        idx = 0
        for v2c in v2c_list:
            print("Experiment: {} # Individual: {} # Best volume to capacity:
                {}".
                  format(idx+1, v2c[0], v2c[1]))
            v2c_csv.append([idx+1, v2c[0], v2c[1]])
            idx += 1
        with open('data/v2c_' + datetime.now().strftime('%m_%d_%Y-%H:%M:%S')
            + '.csv', 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(["Experiment", "Individual", "V2C Ratio"])
            writer.writerows(v2c_csv)

        # Plotting
        data = np.genfromtxt(lb_url, delimiter=",", names=["gen", "nevals",
            "mean"])
        x = np.arange(0, len(data["gen"]), 1)
        plot.figure(0)
        plot.plot(x, data["mean"], color='blue')
        plot.xlabel("Generations")
        plot.ylabel("Mean of Fitness")
        plot.savefig('Nets/SimpleNet/plots/test-' +
            datetime.now().strftime('%m_%d_%Y-%H:%M:%S') + '.png')

        plot.figure(1)
        plot.xlabel("Individuals tested")
        plot.ylabel("MA50 of jam length values")
        for lane in range(0, len(det_ids_list)):
            # Compute the moving average per 50 plans
            jam_ma = gatraffictoolbox.ma(list(zip(*fit_list))[lane], 50)
            plot.plot(np.arange(0, len(jam_ma), 1), jam_ma,
                color="C{}".format(lane))
        plot.savefig('Nets/SimpleNet/plots/jam_test-' +
            datetime.now().strftime('%m_%d_%Y-%H:%M:%S') + '.png')

        print("\nThe initial time was: {}".format(initial_time))
        print("The final time is: {}".format(datetime.now()))
    finally:
        # Restore original xml demand data file
        with open("Nets/SimpleNet/testdemandpedestrian.rou.xml", 'w') as
            original:
            tmp = open(xml_copy.name, 'r')
            original.write(str(tmp.read()))
```

```
if __name__ == "__main__":
    main()
```

B.2.5 trafficinteract.py

This is the script used to create the traffic environments, and it is the an important part of the implementation because is in charge of interact with the environment,retrieving the jam length reawrds an computing the fitness for each individual sent to the simulation, below is depicted the code of the getscore function, which acts as the fitness function, this function is a method of the class TrafficEnv.

```
    def get_score(self, individual, env: tlenvironment.SimulationEnv):
        verbose = True
        ctr = 0
        ts = env.reset()
        array = np.array((self.n_phases, ), dtype=np.float32)
        paramstorage.set_offset("Nets/SimpleNet/tls.xml",
            individual[len(individual) - 1])
        # Always sends the same individual to the simulation for all steps
            of the episode
        for ctr in range(self.n_steps):
            ts = env.step(individual)
            if ts.step_type is array:
                break
            ctr += 1
        scores = list()
        mean = np.mean(ts.reward)
        std = np.std(ts.reward)
        div = np.abs(np.sum(np.gradient(ts.reward)))
        if self.single:
            scores.append(np.mean([mean, std, div]))
        else:
            scores.append(np.round(mean))
            scores.append(np.round(std))
            scores.append(np.round(div))
        tuple(scores)
        if verbose:
            print("\n\033[92mThe individual for this iteration is: ",
                individual)
            print("The jam length for this individual is:
                {}".format(ts.reward))
            print('The score for this individual is:
                {}\033[0m'.format(scores))
        return scores
```

B.2.6 gatraffictools

Into this file we find the MA50 calculator function and the function used to adapt the individuals to a fixed cycle length value. The code of the former function is shown below.

```
def adapt(individual, cycle, lower_limit, upper_limit):
    length = len(individual)-1
```

```
div = []
arr = individual[0:length]
for n in range(0, length):
    div.append(np.floor(np.abs(cycle-np.sum(arr)) / length))
remainder = np.abs(cycle-np.sum(arr)) % length
if cycle-np.sum(arr) > 0:
    for n in range(0, length):
        while arr[n]+div[n] > upper_limit:
            div[n] -= 1
            remainder += 1
        arr[n] += div[n]
    minimum = np.min(arr)
    indices = [i for i, x in enumerate(arr) if x == minimum]
    idx = random.randint(0, len(indices)-1)
    arr[indices[idx]] += remainder
elif cycle - np.sum(arr) < 0:
    for n in range(0, length):
        while arr[n]-div[n] < lower_limit:
            div[n] -= 1
            remainder += 1
        arr[n] -= div[n]
    maximum = np.max(arr)
    indices = [i for i, x in enumerate(arr) if x == maximum]
    idx = random.randint(0, len(indices) - 1)
    arr[indices[idx]] -= remainder
arr.append(individual[length])
return arr
```

B.2.7 sumoconnector.py

Sumo connector act as an interface between the gatraffic application scripts and the sumo simulator, it is in charge of run the sumo program, control the simulation and query it.

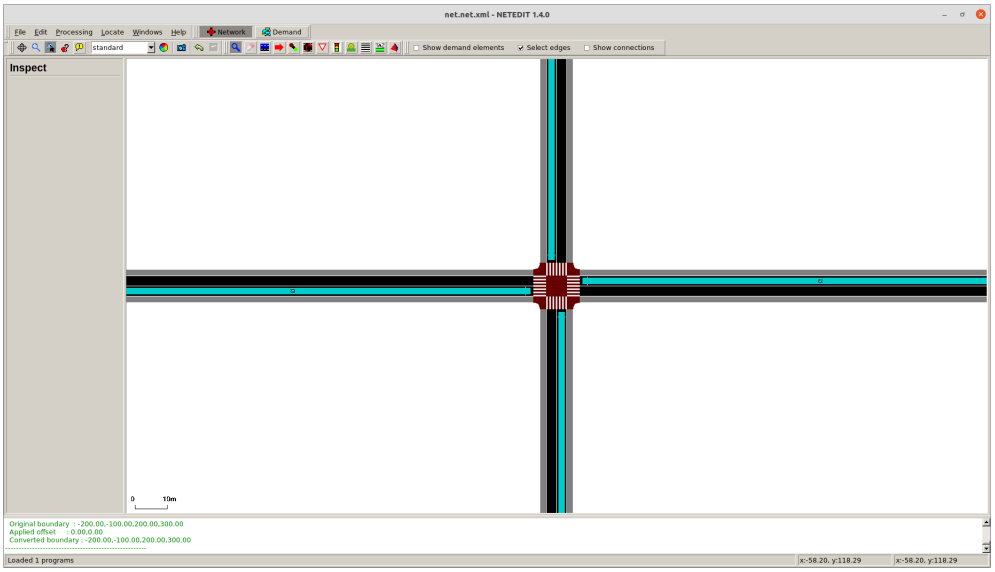
Appendix C

Simulation Environment

C.1 Netedit

The netedit tool included in the sumo package is used to create the simulation environment. It is a GUI to easily create traffic networks. The environment configuration files in this work have been created using this tool.

FIGURE C.1: Netedit tool depicting the simulated environment



C.2 Traffic light editor

Into the netedit tool we also have a editor to create the TL logic, its use is very simple, letting us add new phases and delete it defining the colors of the lights along with other features.

C.3 Sumo-gui

SUMO simulator provides a powerful graphic simulator that let us see interactively the simulation evolution. It is useful when we are performing a single simulation execution, in our case, the overload caused by the repeated executions discourage its use.

FIGURE C.2: Traffic Light Editor

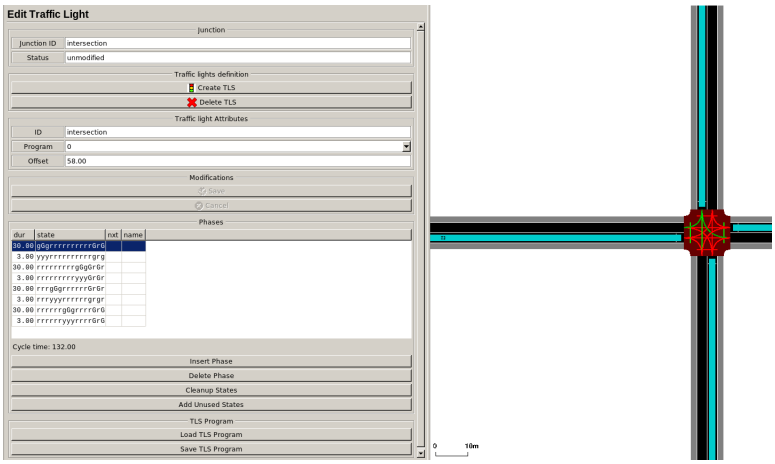


FIGURE C.3: SUMO GUI

C.4 Generated files

Using the SUMO tools we generate several XML files with the configuration content, along this section I am going to show its content.

C.4.1 sumo.sumocfg

This is the file provided when the sumo command is invoked, the files to be loaded are entered into this file, as we see in the xml code below, the network, the demand and the tls file absolute routes are added.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on sat 27 feb 2021 22:17:59 by Eclipse SUMO GUI Version 1.4.0
-->

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">
```

```
    <input>
        <net-file
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/net.net.xml"/>
        <route-files
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/demand_w_ped.rou.xml"/>
        <additional-files
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/additional.add.xml"/>
    </input>

    <gui_only>
        <registry-viewport value="true"/>
        <start value="true"/>
    </gui_only>

</configuration>
```

C.4.2 test-sumo.sumocfg

This is the config file used when the test program is used, in this case the difference comes from the fact that I use a different demand file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on sat 27 feb 2021 22:17:59 by Eclipse SUMO GUI Version 1.4.0
-->

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

    <input>
        <net-file
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/net.net.xml"/>
        <route-files
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/testdemandpedestrian.rou.
        <additional-files
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/additional.add.xml"/>
    </input>

    <gui_only>
        <registry-viewport value="true"/>
        <start value="true"/>
    </gui_only>

</configuration>
```

C.4.3 net.net.xml

The net file is used to define the structure of the whole network, here the roads,the lanes and many other parameters are defined, the generation process, as has stated before, is automated using the netedit program.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- generated on vie 23 jul 2021 19:48:49 by Eclipse SUMO netedit Version
    1.4.0
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/netconvertConfiguration.xsd">

    <input>
        <sumo-net-file
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/net.net.xml"/>
    </input>

    <output>
        <output-file
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/net.net.xml"/>
    </output>

    <processing>
        <geometry.min-radius.fix.railways value="false"/>
        <geometry.max-grade.fix value="false"/>
        <offset.disable-normalization value="true"/>
        <lefthand value="false"/>
    </processing>

    <junctions>
        <no-internal-links value="false"/>
        <no-turnarounds value="true"/>
        <junctions.corner-detail value="5"/>
        <junctions.limit-turn-speed value="5.5"/>
        <rectangular-lane-cut value="false"/>
    </junctions>

    <pedestrian>
        <walkingareas value="false"/>
    </pedestrian>

    <netedit>
        <TLSPrograms-output
            value="/home/miguel/PycharmProjects/GA_Traffic/Nets/SimpleNet/tls.xml"/>
    </netedit>

    <report>
        <aggregate-warnings value="5"/>
    </report>

</configuration>
-->

<net version="1.3" junctionCornerDetail="5" limitTurnSpeed="5.50"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">

    <location netOffset="0.00,0.00"
        convBoundary="-200.00,-100.00,200.00,300.00"
        origBoundary="-10000000000.00,-10000000000.00,10000000000.00,10000000000.00"
        projParameter="!"/>

    <edge id=":east_w0" function="walkingarea">
```

```
    <lane id=":east_w0_0" index="0" allow="pedestrian" speed="1.00"
        length="8.40" width="2.00" shape="200.00,103.20 200.00,105.20
        200.00,94.80 200.00,96.80"/>
</edge>
<edge id=":intersection_0" function="internal">
    <lane id=":intersection_0_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.75" shape="-1.60,107.20 -1.95,104.75
        -3.00,103.00 -3.20,102.88"/>
</edge>
<edge id=":intersection_1" function="internal">
    <lane id=":intersection_1_0" index="0" disallow="pedestrian"
        speed="13.89" length="14.40" shape="-1.60,107.20 -1.60,92.80"/>
</edge>
<edge id=":intersection_2" function="internal">
    <lane id=":intersection_2_0" index="0" disallow="pedestrian"
        speed="8.00" length="10.13" shape="-1.60,107.20 -1.05,103.35
        0.60,100.60 3.20,99.04"/>
</edge>
<edge id=":intersection_12" function="internal">
    <lane id=":intersection_12_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.28" shape="-3.20,102.88 -4.75,101.95
        -7.20,101.60"/>
</edge>
<edge id=":intersection_13" function="internal">
    <lane id=":intersection_13_0" index="0" disallow="pedestrian"
        speed="8.00" length="4.06" shape="3.20,99.04 3.35,98.95
        7.20,98.40"/>
</edge>
<edge id=":intersection_3" function="internal">
    <lane id=":intersection_3_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.75" shape="7.20,101.60 4.75,101.95
        3.00,103.00 2.88,103.20"/>
</edge>
<edge id=":intersection_4" function="internal">
    <lane id=":intersection_4_0" index="0" disallow="pedestrian"
        speed="13.89" length="14.40" shape="7.20,101.60 -7.20,101.60"/>
</edge>
<edge id=":intersection_5" function="internal">
    <lane id=":intersection_5_0" index="0" disallow="pedestrian"
        speed="8.00" length="10.13" shape="7.20,101.60 3.35,101.05
        0.60,99.40 -0.96,96.80"/>
</edge>
<edge id=":intersection_14" function="internal">
    <lane id=":intersection_14_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.28" shape="2.88,103.20 1.95,104.75
        1.60,107.20"/>
</edge>
<edge id=":intersection_15" function="internal">
    <lane id=":intersection_15_0" index="0" disallow="pedestrian"
        speed="8.00" length="4.06" shape="-0.96,96.80 -1.05,96.65
        -1.60,92.80"/>
</edge>
<edge id=":intersection_6" function="internal">
    <lane id=":intersection_6_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.75" shape="1.60,92.80 1.95,95.25
        3.00,97.00 3.20,97.12"/>
</edge>
```

```
<edge id=":intersection_7" function="internal">
    <lane id=":intersection_7_0" index="0" disallow="pedestrian"
        speed="13.89" length="14.40" shape="1.60,92.80 1.60,107.20"/>
</edge>
<edge id=":intersection_8" function="internal">
    <lane id=":intersection_8_0" index="0" disallow="pedestrian"
        speed="8.00" length="10.13" shape="1.60,92.80 1.05,96.65
        -0.60,99.40 -3.20,100.96"/>
</edge>
<edge id=":intersection_16" function="internal">
    <lane id=":intersection_16_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.28" shape="3.20,97.12 4.75,98.05
        7.20,98.40"/>
</edge>
<edge id=":intersection_17" function="internal">
    <lane id=":intersection_17_0" index="0" disallow="pedestrian"
        speed="8.00" length="4.06" shape="-3.20,100.96 -3.35,101.05
        -7.20,101.60"/>
</edge>
<edge id=":intersection_9" function="internal">
    <lane id=":intersection_9_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.75" shape="-7.20,98.40 -4.75,98.05
        -3.00,97.00 -2.88,96.80"/>
</edge>
<edge id=":intersection_10" function="internal">
    <lane id=":intersection_10_0" index="0" disallow="pedestrian"
        speed="13.89" length="14.40" shape="-7.20,98.40 7.20,98.40"/>
</edge>
<edge id=":intersection_11" function="internal">
    <lane id=":intersection_11_0" index="0" disallow="pedestrian"
        speed="8.00" length="10.13" shape="-7.20,98.40 -3.35,98.95
        -0.60,100.60 0.96,103.20"/>
</edge>
<edge id=":intersection_18" function="internal">
    <lane id=":intersection_18_0" index="0" disallow="pedestrian"
        speed="6.51" length="4.28" shape="-2.88,96.80 -1.95,95.25
        -1.60,92.80"/>
</edge>
<edge id=":intersection_19" function="internal">
    <lane id=":intersection_19_0" index="0" disallow="pedestrian"
        speed="8.00" length="4.06" shape="0.96,103.20 1.05,103.35
        1.60,107.20"/>
</edge>
<edge id=":intersection_c0" function="crossing" crossingEdges="IN NI">
    <lane id=":intersection_c0_0" index="0" allow="pedestrian"
        speed="1.00" length="6.40" width="4.00" shape="3.20,105.20
        -3.20,105.20"/>
</edge>
<edge id=":intersection_c1" function="crossing" crossingEdges="IE EI">
    <lane id=":intersection_c1_0" index="0" allow="pedestrian"
        speed="1.00" length="6.40" width="4.00" shape="5.20,96.80
        5.20,103.20"/>
</edge>
<edge id=":intersection_c2" function="crossing" crossingEdges="IS SI">
    <lane id=":intersection_c2_0" index="0" allow="pedestrian"
        speed="1.00" length="6.40" width="4.00" shape="-3.20,94.80
        3.20,94.80"/>
```

```
</edge>
<edge id=":intersection_c3" function="crossing" crossingEdges="IW WI">
    <lane id=":intersection_c3_0" index="0" allow="pedestrian"
        speed="1.00" length="6.40" width="4.00" shape="-5.20,103.20
        -5.20,96.80"/>
</edge>
<edge id=":intersection_w0" function="walkingarea">
    <lane id=":intersection_w0_0" index="0" allow="pedestrian"
        speed="1.00" length="3.30" width="4.00" shape="-7.20,103.20
        -7.20,105.20 -5.20,107.20 -3.20,107.20 -3.31,105.98 -3.64,104.98
        -4.20,104.20 -4.98,103.64 -5.98,103.31"/>
</edge>
<edge id=":intersection_w1" function="walkingarea">
    <lane id=":intersection_w1_0" index="0" allow="pedestrian"
        speed="1.00" length="3.30" width="4.00" shape="3.20,107.20
        5.20,107.20 7.20,105.20 7.20,103.20 5.98,103.31 4.98,103.64
        4.20,104.20 3.64,104.98 3.31,105.98"/>
</edge>
<edge id=":intersection_w2" function="walkingarea">
    <lane id=":intersection_w2_0" index="0" allow="pedestrian"
        speed="1.00" length="3.30" width="4.00" shape="7.20,96.80
        7.20,94.80 5.20,92.80 3.20,92.80 3.31,94.02 3.64,95.02
        4.20,95.80 4.98,96.36 5.98,96.69"/>
</edge>
<edge id=":intersection_w3" function="walkingarea">
    <lane id=":intersection_w3_0" index="0" allow="pedestrian"
        speed="1.00" length="3.30" width="4.00" shape="-3.20,92.80
        -5.20,92.80 -7.20,94.80 -7.20,96.80 -5.98,96.69 -4.98,96.36
        -4.20,95.80 -3.64,95.02 -3.31,94.02"/>
</edge>
<edge id=":north_w0" function="walkingarea">
    <lane id=":north_w0_0" index="0" allow="pedestrian" speed="1.00"
        length="8.40" width="2.00" shape="-3.20,300.00 -5.20,300.00
        5.20,300.00 3.20,300.00"/>
</edge>
<edge id=":south_w0" function="walkingarea">
    <lane id=":south_w0_0" index="0" allow="pedestrian" speed="1.00"
        length="8.40" width="2.00" shape="3.20,-100.00 5.20,-100.00
        -5.20,-100.00 -3.20,-100.00"/>
</edge>
<edge id=":west_w0" function="walkingarea">
    <lane id=":west_w0_0" index="0" allow="pedestrian" speed="1.00"
        length="8.40" width="2.00" shape="-200.00,96.80 -200.00,94.80
        -200.00,105.20 -200.00,103.20"/>
</edge>

<edge id="EI" from="east" to="intersection" priority="-1">
    <lane id="EI_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="200.00,104.20 7.20,104.20"/>
    <lane id="EI_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="200.00,101.60 7.20,101.60"/>
</edge>
<edge id="IE" from="intersection" to="east" priority="-1">
    <lane id="IE_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="7.20,95.80 200.00,95.80"/>
    <lane id="IE_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="7.20,98.40 200.00,98.40"/>
```

```
</edge>
<edge id="IN" from="intersection" to="north" priority="-1">
    <lane id="IN_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="4.20,107.20 4.20,300.00"/>
    <lane id="IN_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="1.60,107.20 1.60,300.00"/>
</edge>
<edge id="IS" from="intersection" to="south" priority="-1">
    <lane id="IS_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="-4.20,92.80 -4.20,-100.00"/>
    <lane id="IS_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="-1.60,92.80 -1.60,-100.00"/>
</edge>
<edge id="IW" from="intersection" to="west" priority="-1">
    <lane id="IW_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="-7.20,104.20
        -200.00,104.20"/>
    <lane id="IW_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="-7.20,101.60 -200.00,101.60"/>
</edge>
<edge id="NI" from="north" to="intersection" priority="-1">
    <lane id="NI_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="-4.20,300.00 -4.20,107.20"/>
    <lane id="NI_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="-1.60,300.00 -1.60,107.20"/>
</edge>
<edge id="SI" from="south" to="intersection" priority="-1">
    <lane id="SI_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="4.20,-100.00 4.20,92.80"/>
    <lane id="SI_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="1.60,-100.00 1.60,92.80"/>
</edge>
<edge id="WI" from="west" to="intersection" priority="-1">
    <lane id="WI_0" index="0" allow="pedestrian" speed="13.89"
        length="192.80" width="2.00" shape="-200.00,95.80 -7.20,95.80"/>
    <lane id="WI_1" index="1" disallow="pedestrian" speed="13.89"
        length="192.80" shape="-200.00,98.40 -7.20,98.40"/>
</edge>

<tlLogic id="intersection" type="static" programID="0" offset="38">
    <phase duration="30" state="gGgrrrrrrrrrrGrG"/>
    <phase duration="3" state="yyyrrrrrrrrrrgrg"/>
    <phase duration="30" state="rrrrrrrrrgGgGrGr"/>
    <phase duration="3" state="rrrrrrrrryyyGrGr"/>
    <phase duration="30" state="rrrgGgrrrrrrGrGr"/>
    <phase duration="3" state="rrryyyrrrrrrgrgr"/>
    <phase duration="30" state="rrrrrrgGgrrrrGrG"/>
    <phase duration="3" state="rrrrrryyyrrrrGrG"/>
</tlLogic>

<junction id="east" type="dead_end" x="200.00" y="100.00" incLanes="IE_0
    IE_1" intLanes="" shape="200.00,100.00 200.00,94.80 200.00,100.00"/>
```

```
<junction id="intersection" type="traffic_light" x="0.00" y="100.00"
    incLanes="NI_0 NI_1 EI_0 EI_1 SI_0 SI_1 WI_0 WI_1 :intersection_w1_0
    :intersection_w2_0 :intersection_w3_0 :intersection_w0_0"
    intLanes=":intersection_12_0 :intersection_1_0 :intersection_13_0
    :intersection_14_0 :intersection_4_0 :intersection_15_0
    :intersection_16_0 :intersection_7_0 :intersection_17_0
    :intersection_18_0 :intersection_10_0 :intersection_19_0
    :intersection_c0_0 :intersection_c1_0 :intersection_c2_0
    :intersection_c3_0" shape="-5.20,107.20 5.20,107.20 5.42,106.09
    5.70,105.70 6.09,105.42 6.59,105.26 7.20,105.20 7.20,94.80
    6.09,94.58 5.70,94.30 5.42,93.91 5.26,93.41 5.20,92.80 -5.20,92.80
    -5.42,93.91 -5.70,94.30 -6.09,94.58 -6.59,94.74 -7.20,94.80
    -7.20,105.20 -6.09,105.42 -5.70,105.70 -5.42,106.09 -5.26,106.59">
    <request index="0" response="1001000100000000"
        foes="1001000100010000" cont="1"/>
    <request index="1" response="0101101100100000"
        foes="0101111100110000" cont="0"/>
    <request index="2" response="0011100011100000"
        foes="0011110011110000" cont="1"/>
    <request index="3" response="0011100010000000"
        foes="0011100010000000" cont="1"/>
    <request index="4" response="1010100110000111"
        foes="1010100110000111" cont="0"/>
    <request index="5" response="0110011110000110"
        foes="0110011110000110" cont="1"/>
    <request index="6" response="0110000000000100"
        foes="0110010000000100" cont="1"/>
    <request index="7" response="0101100000101100"
        foes="0101110000111100" cont="0"/>
    <request index="8" response="1100100000100011"
        foes="1100110000110011" cont="1"/>
    <request index="9" response="1100000000100010"
        foes="1100000000100010" cont="1"/>
    <request index="10" response="1010000111100110"
        foes="1010000111100110" cont="0"/>
    <request index="11" response="1001000110011110"
        foes="1001000110011110" cont="1"/>
    <request index="12" response="0000000000000000"
        foes="0000100010001111" cont="0"/>
    <request index="13" response="0000000000000000"
        foes="0000010001111100" cont="0"/>
    <request index="14" response="0000000000000000"
        foes="0000001111100010" cont="0"/>
    <request index="15" response="0000000000000000"
        foes="0000111100010001" cont="0"/>
</junction>
<junction id="north" type="dead_end" x="0.00" y="300.00" incLanes="IN_0
    IN_1" intLanes="" shape="0.00,300.00 5.20,300.00 0.00,300.00"/>
<junction id="south" type="dead_end" x="0.00" y="-100.00" incLanes="IS_0
    IS_1" intLanes="" shape="0.00,-100.00 -5.20,-100.00 0.00,-100.00"/>
<junction id="west" type="dead_end" x="-200.00" y="100.00"
    incLanes="IW_0 IW_1" intLanes="" shape="-200.00,100.00
    -200.00,105.20 -200.00,100.00"/>

<junction id=":intersection_12_0" type="internal" x="-3.20" y="102.88"
    incLanes=":intersection_0_0" intLanes=":intersection_4_0
    :intersection_8_0 :intersection_c0_0 :intersection_c3_0"/>
```

```
<junction id=":intersection_13_0" type="internal" x="3.20" y="99.04"
    incLanes=":intersection_2_0" intLanes=":intersection_4_0
    :intersection_5_0 :intersection_6_0 :intersection_7_0
    :intersection_10_0 :intersection_11_0 :intersection_c0_0
    :intersection_c1_0"/>
<junction id=":intersection_14_0" type="internal" x="2.88" y="103.20"
    incLanes=":intersection_3_0" intLanes=":intersection_7_0
    :intersection_11_0 :intersection_c0_0 :intersection_c1_0"/>
<junction id=":intersection_15_0" type="internal" x="-0.96" y="96.80"
    incLanes=":intersection_5_0" intLanes=":intersection_1_0
    :intersection_2_0 :intersection_7_0 :intersection_8_0
    :intersection_9_0 :intersection_10_0 :intersection_c1_0
    :intersection_c2_0"/>
<junction id=":intersection_16_0" type="internal" x="3.20" y="97.12"
    incLanes=":intersection_6_0" intLanes=":intersection_2_0
    :intersection_10_0 :intersection_c1_0 :intersection_c2_0"/>
<junction id=":intersection_17_0" type="internal" x="-3.20" y="100.96"
    incLanes=":intersection_8_0" intLanes=":intersection_0_0
    :intersection_1_0 :intersection_4_0 :intersection_5_0
    :intersection_10_0 :intersection_11_0 :intersection_c2_0
    :intersection_c3_0"/>
<junction id=":intersection_18_0" type="internal" x="-2.88" y="96.80"
    incLanes=":intersection_9_0" intLanes=":intersection_1_0
    :intersection_5_0 :intersection_c2_0 :intersection_c3_0"/>
<junction id=":intersection_19_0" type="internal" x="0.96" y="103.20"
    incLanes=":intersection_11_0" intLanes=":intersection_1_0
    :intersection_2_0 :intersection_3_0 :intersection_4_0
    :intersection_7_0 :intersection_8_0 :intersection_c0_0
    :intersection_c3_0"/>

<connection from="EI" to="IN" fromLane="1" toLane="1"
    via=":intersection_3_0" tl="intersection" linkIndex="3" dir="r"
    state="o"/>
<connection from="EI" to="IW" fromLane="1" toLane="1"
    via=":intersection_4_0" tl="intersection" linkIndex="4" dir="s"
    state="o"/>
<connection from="EI" to="IS" fromLane="1" toLane="1"
    via=":intersection_5_0" tl="intersection" linkIndex="5" dir="l"
    state="o"/>
<connection from="NI" to="IW" fromLane="1" toLane="1"
    via=":intersection_0_0" tl="intersection" linkIndex="0" dir="r"
    state="o"/>
<connection from="NI" to="IS" fromLane="1" toLane="1"
    via=":intersection_1_0" tl="intersection" linkIndex="1" dir="s"
    state="o"/>
<connection from="NI" to="IE" fromLane="1" toLane="1"
    via=":intersection_2_0" tl="intersection" linkIndex="2" dir="l"
    state="o"/>
<connection from="SI" to="IE" fromLane="1" toLane="1"
    via=":intersection_6_0" tl="intersection" linkIndex="6" dir="r"
    state="o"/>
<connection from="SI" to="IN" fromLane="1" toLane="1"
    via=":intersection_7_0" tl="intersection" linkIndex="7" dir="s"
    state="o"/>
<connection from="SI" to="IW" fromLane="1" toLane="1"
    via=":intersection_8_0" tl="intersection" linkIndex="8" dir="l"
    state="o"/>
```

```
<connection from="WI" to="IS" fromLane="1" toLane="1"
    via=":intersection_9_0" tl="intersection" linkIndex="9" dir="r"
    state="o"/>
<connection from="WI" to="IE" fromLane="1" toLane="1"
    via=":intersection_10_0" tl="intersection" linkIndex="10" dir="s"
    state="o"/>
<connection from="WI" to="IN" fromLane="1" toLane="1"
    via=":intersection_11_0" tl="intersection" linkIndex="11" dir="l"
    state="o"/>

<connection from=":intersection_0" to="IW" fromLane="0" toLane="1"
    via=":intersection_12_0" dir="r" state="m"/>
<connection from=":intersection_12" to="IW" fromLane="0" toLane="1"
    dir="r" state="M"/>
<connection from=":intersection_1" to="IS" fromLane="0" toLane="1"
    dir="s" state="M"/>
<connection from=":intersection_2" to="IE" fromLane="0" toLane="1"
    via=":intersection_13_0" dir="l" state="m"/>
<connection from=":intersection_13" to="IE" fromLane="0" toLane="1"
    dir="l" state="M"/>
<connection from=":intersection_3" to="IN" fromLane="0" toLane="1"
    via=":intersection_14_0" dir="r" state="m"/>
<connection from=":intersection_14" to="IN" fromLane="0" toLane="1"
    dir="r" state="M"/>
<connection from=":intersection_4" to="IW" fromLane="0" toLane="1"
    dir="s" state="M"/>
<connection from=":intersection_5" to="IS" fromLane="0" toLane="1"
    via=":intersection_15_0" dir="l" state="m"/>
<connection from=":intersection_15" to="IS" fromLane="0" toLane="1"
    dir="l" state="M"/>
<connection from=":intersection_6" to="IE" fromLane="0" toLane="1"
    via=":intersection_16_0" dir="r" state="m"/>
<connection from=":intersection_16" to="IE" fromLane="0" toLane="1"
    dir="r" state="M"/>
<connection from=":intersection_7" to="IN" fromLane="0" toLane="1"
    dir="s" state="M"/>
<connection from=":intersection_8" to="IW" fromLane="0" toLane="1"
    via=":intersection_17_0" dir="l" state="m"/>
<connection from=":intersection_17" to="IW" fromLane="0" toLane="1"
    dir="l" state="M"/>
<connection from=":intersection_9" to="IS" fromLane="0" toLane="1"
    via=":intersection_18_0" dir="r" state="m"/>
<connection from=":intersection_18" to="IS" fromLane="0" toLane="1"
    dir="r" state="M"/>
<connection from=":intersection_10" to="IE" fromLane="0" toLane="1"
    dir="s" state="M"/>
<connection from=":intersection_11" to="IN" fromLane="0" toLane="1"
    via=":intersection_19_0" dir="l" state="m"/>
<connection from=":intersection_19" to="IN" fromLane="0" toLane="1"
    dir="l" state="M"/>

<connection from=":east_w0" to="EI" fromLane="0" toLane="0" dir="s"
    state="M"/>
<connection from="IE" to=":east_w0" fromLane="0" toLane="0" dir="s"
    state="M"/>
<connection from=":intersection_c0" to=":intersection_w0" fromLane="0"
    toLane="0" dir="s" state="M"/>
```

```
    <connection from=":intersection_c1" to=":intersection_w1" fromLane="0"
        toLane="0" dir="s" state="M"/>
    <connection from=":intersection_c2" to=":intersection_w2" fromLane="0"
        toLane="0" dir="s" state="M"/>
    <connection from=":intersection_c3" to=":intersection_w3" fromLane="0"
        toLane="0" dir="s" state="M"/>
    <connection from=":intersection_w0" to=":intersection_c3" fromLane="0"
        toLane="0" tl="intersection" linkIndex="15" dir="s" state="M"/>
    <connection from=":intersection_w0" to="IW" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from="NI" to=":intersection_w0" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from=":intersection_w1" to=":intersection_c0" fromLane="0"
        toLane="0" tl="intersection" linkIndex="12" dir="s" state="M"/>
    <connection from=":intersection_w1" to="IN" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from="EI" to=":intersection_w1" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from=":intersection_w2" to=":intersection_c1" fromLane="0"
        toLane="0" tl="intersection" linkIndex="13" dir="s" state="M"/>
    <connection from=":intersection_w2" to="IE" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from="SI" to=":intersection_w2" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from=":intersection_w3" to=":intersection_c2" fromLane="0"
        toLane="0" tl="intersection" linkIndex="14" dir="s" state="M"/>
    <connection from=":intersection_w3" to="IS" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from="WI" to=":intersection_w3" fromLane="0" toLane="0"
        dir="s" state="M"/>
    <connection from=":north_w0" to="NI" fromLane="0" toLane="0" dir="s"
        state="M"/>
    <connection from="IN" to=":north_w0" fromLane="0" toLane="0" dir="s"
        state="M"/>
    <connection from=":south_w0" to="SI" fromLane="0" toLane="0" dir="s"
        state="M"/>
    <connection from="IS" to=":south_w0" fromLane="0" toLane="0" dir="s"
        state="M"/>
    <connection from=":west_w0" to="WI" fromLane="0" toLane="0" dir="s"
        state="M"/>
    <connection from="IW" to=":west_w0" fromLane="0" toLane="0" dir="s"
        state="M"/>
</net>
```

C.4.4 demandpedestrian.rou.xm

The demand file is important since we define the flows into it, I have created two types of flows, vehicles and persons. Here we also define the time the simulation lasts, since it depends the time the flows exists.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on lun 29 mar 2021 21:09:47 by Eclipse SUMO netedit Version
    1.4.0
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/netconvertConfiguration.xsd">
```

```
    <input>
        <sumo-net-file
            value="/home/miguel/PycharmProjects/DRL_Traffic/Nets/SimpleNet/net.net.xml"/>
    </input>

    <output>
        <output-file
            value="/home/miguel/PycharmProjects/DRL_Traffic/Nets/SimpleNet/net.net.xml"/>
    </output>

    <processing>
        <geometry.min-radius.fix.railways value="false"/>
        <geometry.max-grade.fix value="false"/>
        <offset.disable-normalization value="true"/>
        <lefthand value="false"/>
    </processing>

    <junctions>
        <no-internal-links value="false"/>
        <no-turnarounds value="true"/>
        <junctions.corner-detail value="5"/>
        <junctions.limit-turn-speed value="5.5"/>
        <rectangular-lane-cut value="false"/>
    </junctions>

    <pedestrian>
        <walkingareas value="false"/>
    </pedestrian>

    <netedit>
        <route-files
            value="/home/miguel/PycharmProjects/DRL_Traffic/Nets/SimpleNet/demand_w_ped.rou.xml"/>
    </netedit>

    <report>
        <aggregate-warnings value="5"/>
    </report>

</configuration>
-->

<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <route edges="EI IN" color="green" id="route_EN"/>
    <route edges="EI IS" color="green" id="route_ES"/>
    <route edges="EI IW" color="green" id="route_EW"/>
    <route edges="NI IE" color="red" id="route_NE"/>
    <route edges="NI IS" color="red" id="route_NS"/>
    <route edges="NI IW" color="red" id="route_NW"/>
    <route edges="SI IE" color="blue" id="route_SE"/>
    <route edges="SI IN" color="blue" id="route_SN"/>
    <route edges="SI IW" color="blue" id="route_SW"/>
    <route edges="WI IE" color="yellow" id="route_WE"/>
    <route edges="WI IN" color="yellow" id="route_WN"/>
    <route edges="WI IS" color="yellow" id="route_WS"/>
```

```
    <flow id="flow_NE" begin="0.00" color="red" from="NI" to="IE"
        end="450.00" probability="0.05"/>
    <flow id="flow_NS" begin="0.00" color="red" from="NI" to="IS"
        end="450.00" probability="0.05"/>
    <flow id="flow_NW" begin="0.00" color="red" from="NI" to="IW"
        end="450.00" probability="0.05"/>
    <flow id="flow_WE" begin="0.00" color="yellow" from="WI" to="IE"
        end="450.00" probability="0.01"/>
    <flow id="flow_WN" begin="0.00" color="yellow" from="WI" to="IN"
        end="450.00" probability="0.06"/>
    <flow id="flow_WS" begin="0.00" color="yellow" from="WI" to="IS"
        end="450.00" probability="0.03"/>
    <flow id="flow_EN" begin="0.00" color="green" from="EI" to="IN"
        end="450.00" probability="0.025"/>
    <flow id="flow_ES" begin="0.00" color="green" from="EI" to="IS"
        end="450.00" probability="0.01"/>
    <flow id="flow_EW" begin="0.00" color="green" from="EI" to="IW"
        end="450.00" probability="0.08"/>
    <flow id="flow_SE" begin="0.00" color="blue" from="SI" to="IE"
        end="450.00" probability="0.05"/>
    <flow id="flow_SN" begin="0.00" color="blue" from="SI" to="IN"
        end="450.00" probability="0.04"/>
    <flow id="flow_SW" begin="0.00" color="blue" from="SI" to="IW"
        end="450.00" probability="0.01"/>
    <personFlow id="personFlow_E" begin="0.00" end="450.00"
        probability="0.10">
        <personTrip from="EI" to="IW"/>
    </personFlow>
    <personFlow id="personFlow_N" begin="0.00" end="450.00"
        probability="0.10">
        <personTrip from="NI" to="IS"/>
    </personFlow>
    <personFlow id="personFlow_S" begin="0.00" end="450.00"
        probability="0.10">
        <personTrip from="SI" to="IN"/>
    </personFlow>
    <personFlow id="personFlow_W" begin="0.00" end="450.00"
        probability="0.10">
        <personTrip from="WI" to="IE"/>
    </personFlow>
</routes>
```

C.4.5 tls.xml

The traffic lights logic is into this file, the letters G and g define, respectively, pass with and without preference,the r symbolize not allowed passing, the y indicates a yellow light.

```
<additionals>
    <tlLogic id="intersection" type="static" programID="0" offset="28">
        <phase duration="30" state="gGgrrrrrrrrrrGrG" />
        <phase duration="3" state="yyyrrrrrrrrrrgrg" />
        <phase duration="3" state="rrrrrrrrrrrrrrrr" />
        <phase duration="30" state="rrrrrrrrrgGgGrGr" />
        <phase duration="3" state="rrrrrrrrryyyGrGr" />
```

```
        <phase duration="3" state="rrrrrrrrrrrrGrGr" />
        <phase duration="30" state="rrrgGgrrrrrrGrGr" />
        <phase duration="3" state="rrryyyrrrrrrgrgr" />
        <phase duration="3" state="rrrrrrrrrrrrrrrr" />
        <phase duration="30" state="rrrrrrgGgrrrrGrG" />
        <phase duration="3" state="rrrrrryyyrrrrGrG" />
        <phase duration="3" state="rrrrrrrrrrrrrGrG" />
    </tlLogic>

</additionals>
```

References

[1] Andrew Hamilton et al. *The evolution of urban traffic control: changing policy and technology*. Feb. 2013. DOI: 10.1080/03081060.2012.745318.

[2] Julio Sanguesa et al. *Sensing Traffic Density Combining V2V and V2I Wireless Communications*. Dec. 2015. DOI: 10.3390/s151229889.

[3] Tanzina Afrin and Nita Yodo. *A Survey of Road Traffic Congestion Measures towards a Sustainable and Resilient Transportation System*. June 2020. DOI: 10.3390/su12114660.

[4] Diamantis Manolis et al. "Centralised vs decentralised signal control of large-scale urban road networks in real time: A simulation study". In: *IET Intelligent Transport Systems* 12 (May 2018). DOI: 10.1049/iet-its.2018.0112.

[5] Deepa S. Sivanandam S. "Genetic Algorithms". In: *Introduction to Genetic Algorithms*. Berlin: Springer, 2018. Chap. Genetic Algorithms, pp. 15–37. ISBN: 978-3-540-73189-4. DOI: 10.1007/978-3-540-73190-0.

[6] Elena Sofronova, A.A. Belyakov, and D.B. Khamadiyarov. "Optimal Control for Traffic Flows in the Urban Road Networks and Its Solution by Variational Genetic Algorithm". In: *Procedia Computer Science* 150 (Jan. 2019), pp. 302–308. DOI: 10.1016/j.procs.2019.02.056.

[7] Samara Soares Leal, Paulo Eduardo Maciel de Almeida, and Edward Chung. "Active control for traffic lights in regions and corridors: an approach based on evolutionary computation". In: *Transportation Research Procedia* 25 (2017). World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016, pp. 1769–1780. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2017.05.140. URL: https://www.sciencedirect.com/science/article/pii/S2352146517304313.

[8] Lin Yilun, Xingyuan Dai, and Li Li. *An Efficient Deep Reinforcement Learning Model for Urban Traffic Control*. Aug. 2018.

[9] A. Vidali et al. "A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management". In: *WOA*. 2019.

[10] Cambridge. *Dictionary*. 2021. URL: https://dictionary.cambridge.org/dictionary/english-spanish/environment.

[11] Eclipse. *Simulation of Urban MObility*. 2021. URL: https://www.eclipse.org/sumo/.

[12] DEAP. *Distributed Evolutionary Algorithms in Python*. 2021. URL: https://deap.readthedocs.io/en/master/.

[13] Tensor Flow. *Tf-Agents*. 2021. URL: https://www.tensorflow.org/agents.

[14] Ioannis Giagkiozis and Peter Fleming. "Pareto Front Estimation for Decision Making". In: *Evolutionary computation* 22 (Apr. 2014). DOI: 10.1162/EVCO_a_00128.

[15] Mahesh Kumar, Perumal Nallagownden, and Irraivan Elamvazuthi. "Advanced Pareto Front Non-Dominated Sorting Multi-Objective Particle Swarm Optimization for Optimal Placement and Sizing of Distributed Generation". In: *Energies* 9 (Nov. 2016), p. 982. DOI: 10.3390/en9120982.

[16] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[17] Ivana Nedevska, Slobodan Ognjenović, and Elena Gusakova. "Methodology for analysing capacity and level of service for signalized intersections (HCM 2000)". In: *MATEC Web of Conferences* 86 (2016), p. 05026. DOI: 10.1051/matecconf/20168605026.