

Altering Traffic Light Timings to Reduce Car Usage

Jesse Tobias Dissel



Universiteit Utrecht

7.5 EC

BA Artificial Intelligence thesis

Student number: 6286518

E-mail: j.t.dissel@students.uu.nl

Supervisor: dr. Tomas Klos

Reader: prof. dr. Hans Bodlaender

Date: 02-07-2021

Abstract

Widespread car usage is a problem that cities worldwide are trying to control. They try to promote other forms of mobilities such as walking, cycling, and using public transport, while punishing car usage by forcing motorized traffic onto longer routes and pricing congestion. With reducing car usage as a goal without clogging up the road, we designed a genetic algorithm that was able to change behavior of agents based on the duration of traffic light timings. By elongating the length of green lights for non-motorized traffic (cyclists and pedestrians) compared to traffic lights for motorized traffic (cars and busses) with 35%, the model was able to improve the percentage of agents using eco-friendly transportation modes with 12.8 percentage points.

Table of contents

1.	Introduction	4
2.	Method.....	6
3.	Model structure.....	11
4.	Results	15
5.	Conclusion & Discussion	19
6.	References	20

1. Introduction

Since the rise of the automobile in the 20th century, cars have been the predominant users of the roads, while cyclists and pedestrians have been mostly forced over to sidewalks and cycle paths. By designing streets and cities for the car, car ownership has spiked to 505 cars per 1.000 inhabitants in the EU (CBS, 2019). Annual milage of all Dutch road traffic reached an all-time high in 2017, while the average trip length of private vehicles fell by 6% compared to the previous year. Consequently, the amount of vehicle ownership has increased (CBS, 2018).

This might not seem like a problem. Mobility is linked to GDP-growth (Beyzaltar et. al, 2014). Businesses use roads to transport goods, and civilians use the roads to go to their jobs and buy goods at shops. However, not all forms of mobility are created equal.

In cities, where space is at a premium, private cars take up a lot of space. When driving a car in an urban environment, a car takes up about 107 square meters (Kwantes, 2019). This comes because the space a car takes up is not only determined by its physical size, but also by the headway it reserves when driving at speed. Since annual milage is still rising, roads are becoming more congested. And although cars are becoming greener since the rise in popularity of hybrid and electric cars, road traffic is still responsible for 19% of all CO₂ emissions in the Netherlands in 2019 (CBS, 2021). On average, driving costs society €0.15/km, while cycling earns society €0.16/km. It is therefore important that car usage needs to be disincentivized.

A lot of cities around the world are already enforcing measures to disincentivize driving a car, while making sure people can move around using other modes of mobility. These measures generally fall in the following two categories: promoting other modes and punishing car usage. Some measures fall strictly in one category, while others overlap.

Promoting other uses

Cities around the world are transforming their streets to accommodate cyclists. In countries like the Netherlands and Denmark they are a common sight and cycling is an everyday activity, but that has not always been the case. These countries started investing in bicycle infrastructure in the second half of the 20th century and the Dutch are reaping the benefits with 26% of all trips being done by bike (Ministerie van Infrastructuur en Waterstaat, 2019).

In other parts of the world, more and more dedicated bicycle infrastructure is popping up in cities such as New York (NYC DOT, n.d.) and Paris (Ile-de-France, 2021). The Covid pandemic has been used by these cities, as well as many others, to temporarily remove car lanes in favor of bicycle lanes or transform streets to pedestrianized streets. It is believed that these transformations will stay when the pandemic is over (France24, 2020).

Cities are also trying to attract more people to public transport. Some cities are reducing the cost of a public transport year ticket. For example, the cost of a public transport card is reduced to 365 euros in what is called the Vienna model (Oltermann, 2019). Other cities like Luxembourg have abolished public transport payments all together (Mobiliteits Zentral, 2020).

Punishing car usage

In most cities, prices for city-center parking are going up, or parking is straight out banned (Parkopedia, 2019). This is an incentive to use public transport or go by bike when going downtown. Some cities like Ghent are banning through traffic and are forcing motorized traffic the long way around, while pedestrians, bicyclists and public transport users can go through (Obordo, 2020). Barcelona uses the ‘supermanzanas/superblock’ concept to transform busy streets to more peaceful and spacious inner areas, while through traffic is again forced the long way around (Energy cities, 2016). Another model used is banning or pricing the most polluting vehicles when entering a city (Sisson, 2020). Streets are also seeing speed reductions, either by introducing traffic-hindering measures such as speedbumps and chicanes, but also reducing the maximum speeds to 30 km/h according to the Vision Zero proposal that aims to reduce the number of traffic related deaths to zero, that has been adopted by various countries (Goodyear, 2014).

Most of the punishing measures aim to increase travel times for those who are driving a car. But another possibility to increase travel times, that does not require infrastructural improvements, does not seem to be part of the discussion. This possibility is changing the waiting times at traffic lights.

Almost all traffic lights are set up in such a way to let the most vehicles or persons through a given intersection. The Jevons paradox (1865) states that an increase in efficiency tends to increase, rather than decrease, the rate of consumption. Given this, we can hypothesize that if we were to optimize flows of public transport vehicles, cyclists, and pedestrians in favor of car traffic, people would be more inclined to use these forms of transit. Consequently, this would mean a decrease in efficiency for car traffic, and thus a reduction in car usage.

RQ: How could traffic light timings be used to reduce car usage?

We divide this research question into the following two sub-questions:

SQ1: Do agents change their behavior such that they use other modes of mobility?

SQ2: Do successful traffic lights punish car drivers with longer waiting times?

Since the option of using anything other than cars is not a possibility for everyone – some people have disabilities, sometimes public transport does not reach, etc. – the focus needs to be on the drivers that have greener alternatives available to them but choose not to use them. Therefore, an optimum has to be sought such that red light signals for drivers are being extended so that as many drivers as possible will sway towards using those other modes of transit, while the road does not jam up entirely for those that have it as their only option.

The way this optimum set of traffic light timings has to be found is by using a heuristic, since the search space is large and complex. We do not know the direct relation between traffic light timings and mobility choices, but the effects can be measured. A heuristic tries to approximate a solution but cannot guarantee that the optimal solution will be found. This is however not a major problem, since we are not per se searching for the optimal set of timings, but an improvement would be satisfiable. Of course, we want to change behavior as much as possible, but perfection is not a requirement.

One way to apply a heuristic could be using a genetic algorithm (GA), that is part of the larger class of evolutionary algorithms. Teo et al. (2010) have used a GA in a model to optimize traffic light timings, while Turky et al. (2009) have used a GA in combination with a Cellular Automata to model and optimize traffic light control optimization for pedestrian crossings.

A GA would be a good candidate to search through the search space, because of its simplicity. It is a method that is relatively easy to implement, and the decisions it makes are well explainable. A GA is inspired by evolution theory and uses a form of natural selection to improve on previous iterations. To determine how good one intersection performs, a fitness function is used. In this case, the goal of the GA would be to optimize the usage of green modes of traffic without causing backups for cars.

SQ3: What does a fitness measure look like so that it considers waiting times and vehicular uses?

SQ4: When do drivers move away from driving a car and instead use other modes of transportation?

If we can answer these last 2 sub-questions, we are able to create a model that allows us to answer our main research question. Then, the model can promote what it determines ‘good’ traffic lights and allows agents to change behavior. The ‘Method’-chapter will further elaborate how these questions are going to be answered.

2. Method

To explore the research questions, we will use the agent-based simulation tools provided by open-source project SUMO (Simulation of Urban MObility). SUMO is a computer program designed by the German Aerospace Center (DLR) in 2001 and sees regular updates. It allows users to simulate all different kinds of mobilities in an urban setting. The urban setting can be modelled to a real-life city or a single intersection. Agents can be modelled to travel from one place to another and can use an assigned mode of transportation. In this model we will allow agents to travel either by foot, bicycle, private car, or by bus.

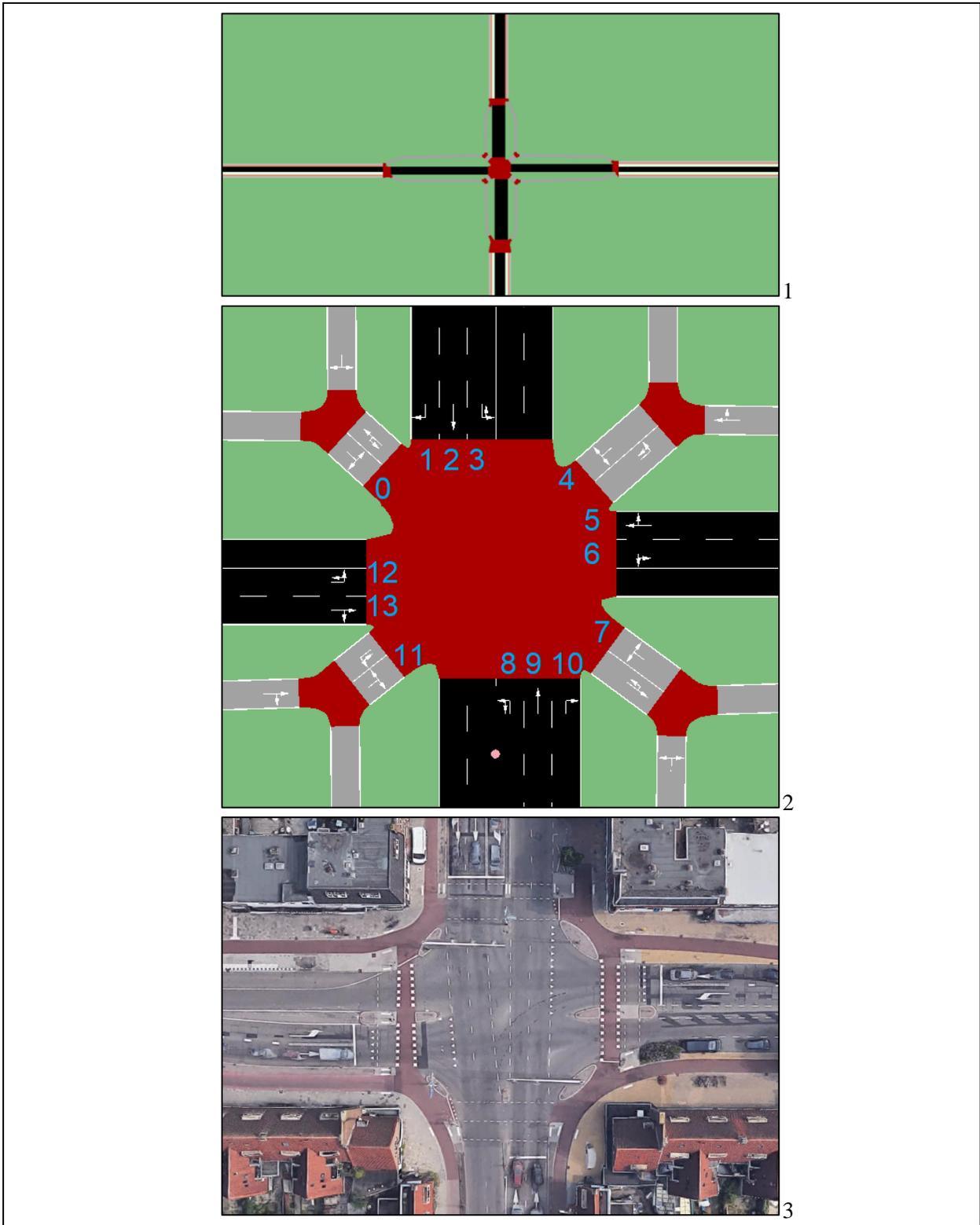
The aim of this research is to see what the effects of a change in traffic light timings are on vehicular uses. We will examine the effects of altering the traffic lights of one intersection, and not focus on a setup that contains multiple intersections. To see the changes in behavior, a single simulation of one selected traffic light setup is not enough, because it does not allow agents to switch to another form of transportation. Therefore, after each simulation agents will reevaluate their mode of transportation. If an agent drove by car in the first simulation and the waiting time was too long, it might go by bike, bus, or walk instead the next time. Likewise, a cyclist that waited for too long at the intersection might change to go by car next time. The total number of agents will be the same for each simulation. In the ‘Model Structure’-chapter we will explore how this is applied to our model.

Intersection in SUMO

To simulate our agents, we will use an intersection that is modeled after a common Dutch intersection. Because there is no standard intersection – all intersections are different given the context of a city layout – some assumptions and generalizations have been made. The designed intersection consists of a four-way crossing with a four lane north-south road and a two lane east-west road. The roads themselves have an extra dedicated turning lane for motorized traffic, and alongside each road are combined walking and bicycle paths, all one way. Direct diagonal crossing for cyclists and pedestrians is not allowed, as it is most often in real life the case. Instead, cyclists and pedestrians must cross the intersection two times, going either clockwise or counterclockwise around the intersection. Major enhancements such as tunnels, overpasses and/or actuators in the asphalt that ‘sense’ when vehicles approach the intersection, have been left out. Intersections that contain such features are less common and have already seen expensive improvements.

The modelled intersection consists of 14 individual traffic signals, with one light for each lane. Because lanes are restricted to either motorized (cars and busses) or non-motorized (pedestrians and cyclists) forms of transportation, we can differentiate between lights that serve these traffic modes. In the model presented below, we see that lights 0, 4, 7, and 11 are signals that are restricted to non-motorized traffic, while the other lights are restricted to motorized traffic only. Some signals allow only one movement. For example, light 2 only allows movements from north to south, while others allow multiple movements. Signals operate independently of each other, and conflicting flows of traffic are allowed. If conflicting movements occur, SUMO gives priority to movements coming from the right side.

Agents will travel randomly through this intersection. They do this by randomly choosing a start and end point for their journey. These start and end points of their journey are either north, south, east, or west, corresponding to the outermost points of that road. To complete a journey, agents must travel through the intersection, where they will travel towards the chosen end point. The way they choose their mode of transportation will be explained in chapter 3.



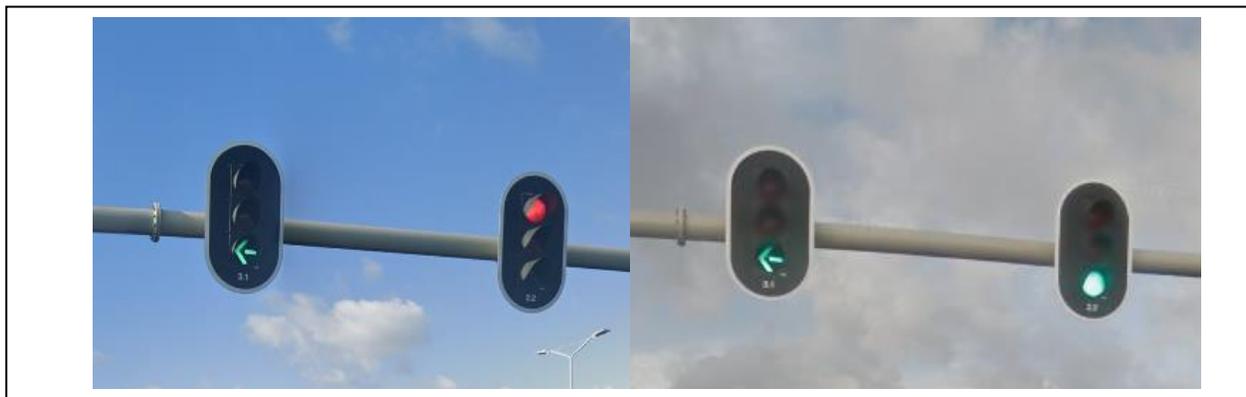
1: The intersection as modelled in SUMO. An east-west road and north-south road cross at the intersection.
 2: A close-up of the intersection. Motorized traffic uses roads, non-motorized traffic uses paths. Signals are labeled.
 3: A similar intersection: the Amsterdamsstraatweg and the Marnixlaan in Utrecht.

Now that we have decided what intersection we will use for our simulation, we will explore how traffic lights work, and how they can be manipulated.

Traffic lights as Finite State Machines

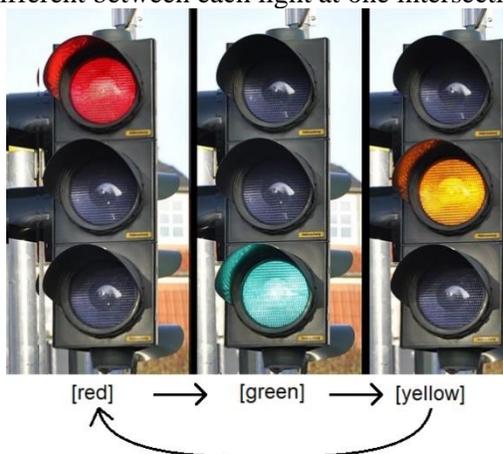
The internal logic used by traffic lights is a form of a *Finite State Machine (FSM)*. An FSM can be in only one state at a time. An FSM exists of three components: a list of states, the initial state, and the inputs that trigger a transition to another state.

We can see that a traffic light uses an FSM for its internal logic. At a given moment, individual lights at an intersection are set-up in a particular way. For example, the state for an intersection consisting of two lights can be at one moment [green, red]. This means that at that state light one is red and light two is green. At a set moment the internal logic of the traffic light system determines that it is time to transition to the next state, for example [green, green]. This means that the list of states in the FSM consists of the members [green, red] and [green, green]. [green, red] is the initial state, and the internal logic of the intersection caused a transition to the state [green, green].



Two states in a traffic light FSM. The moment of transition is determined by the internal logic of the traffic light system.

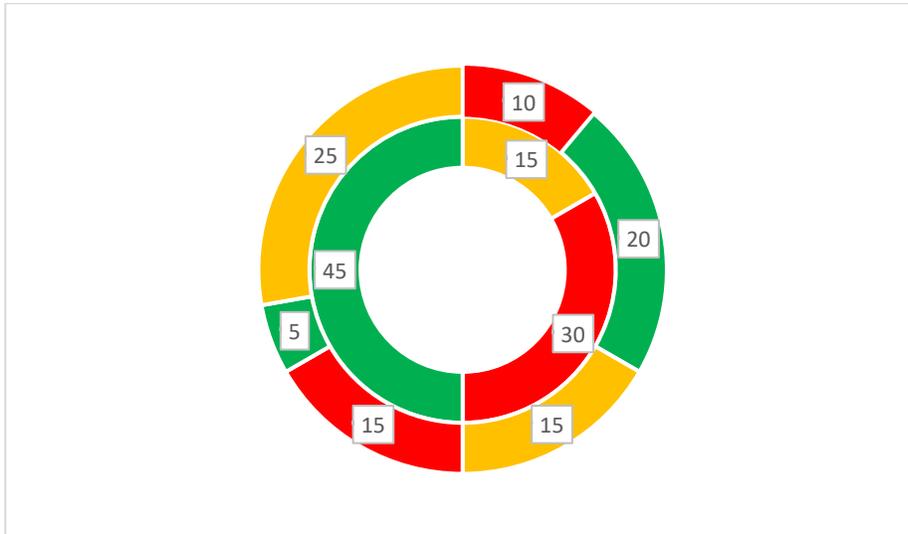
Importantly, each individual light at a given intersection is also an FSM. This FSM for each individual light is always a cycle of 3 states. This is because a red light will always turn green, a green light will always become yellow, and after a given amount of time a yellow light will always turn red. The list of states for each light is therefore always [red, green, yellow]. However, starting states and the logic that causes a light to change can be different between each light at one intersection.



An FSM for one single traffic light. After a state of red, a green will follow, after a state of green a yellow will follow and after a state of yellow a red state will follow.

We can visualize the percentage of time a traffic lights spend on given states as a clock. Each traffic light is one ring of the clock, and the slices of the clock are the different states that traffic light is in. In the example below this means that light 1 (the outermost ring) starts in a 'red' state, and after 10 seconds transitions to a green state. Meanwhile traffic light two starts in a 'yellow' state and transitions after 15 seconds to a 'red' state.

For each ring, the number of slices and their sizes can differ, but they finish at the same moment. This means that the number of states and their timings can vary, but the sum of all timings stays the same. This is important because we want the timings of all traffic lights to line up at the end of the program, so when it is finished it can loop and start over without timings getting out of sync.



A Traffic light program represented as a clock. The rings correspond to two signals at one intersection. The sizes of the slices are labeled and correspond to the duration of the states, with all durations being assigned random values. The program can loop after it is finished because the sum of timings is the same for all lights.

How transitions and timings are implemented are part of the model design and will be explored in the next chapter: 'Model structure'. But first we will explore how a genetic algorithm works, so that we can apply it to our model.

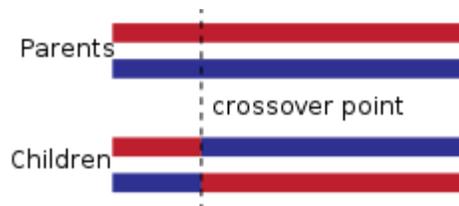
Genetic Algorithm and fitness score

To find a solution in the search space, a Genetic Algorithm is used. A genetic Algorithm is inspired by evolution theory. In the GA, a traffic light program will be encoded into a DNA string. As natural selection is done on group level, multiple DNA strings are grouped in a generation. For each of the DNA-strings we want to know how well it performs when simulating, so they all get a fitness score assigned based on their performance. This fitness score is important because it has a large influence on what members of the generation are allowed to generate DNA strings that will feature in the next generation. The idea is that only the most successful members of a generation will generate offspring. Generating offspring consists of three core components. These are selection, crossover, and mutation. This is an iterative process.

First is selection. As with natural selection, only the strongest members will survive. In this case this means that the DNA strings that have got the highest fitness scores assigned will feature in the pool of parents that are allowed to breed the new generation. There are multiple ways to introduce selection in a simulation, such as with tournaments, in which individuals must compete against other individuals where only those with the highest fitness score go through. Another common method is roulette simulation, where random chances determine the selection. We have chosen to use Truncation Selection. Here, only

the fittest group is allowed to generate offspring, and individuals with a lower fitness score have no chance to generate offspring. This is an easy and efficient method of breeding (Crow & Kimura, 1979).

Next is crossover. Here, DNA of two individuals is selected, and from their DNA offspring is generated. Most often the two parents will generate two children. A crossover chromosome is randomly selected. This is the crossover point at which point the DNA for the children will switch sides. So, for one point crossover, we will see that the DNA string of child 1 will consist of the DNA from parent A up until the crossover point, while the second part from the crossover point to the end will be from parent B's DNA string, and vice versa for child 2.



Single point crossover. Two DNA strings of parents A and B result in different DNA strings for children 1 and 2. (Wikipedia, n.d.)

Lastly there is mutation. One genome of the DNA of a child randomly changes to introduce randomness and diversity. This is important to make sure the Genetic Algorithm will explore alternative options and will not get stuck on a local minimum.

Now that we determined the intersection we will do our simulations on, know how traffic lights work as finite state machines and understand the way a genetic algorithm searches through a search space, we will explore how it has been applied in this simulation.

3. Model structure

What features are needed in the code to simulate an intersection?

To apply a Genetic Algorithm on an intersection, we need to understand what features are needed to make this work. We have seen that a GA uses a pool of DNA-strings and applies selection, crossover, and mutation. To simulate these DNA-strings we will use SUMO, so it is important to see how a traffic light is coded in SUMO.

A traffic light program in SUMO consists of a list of *phases* that loops, like a Finite State Machine. A *phase* consists of two elements: a *duration* in seconds, coded as an integer, and a *state*, a list that contains for each light its color, coded as a string where the first letter corresponds to the color of light 1, the second letter to the color of light 2, and so on. The letter ‘G’ means that a light is green, ‘y’ means that it is yellow, and ‘r’ means that it is red. When a phase begins, a countdown will start from the *duration*-value. After the countdown reaches zero, the next phase will start. At that moment, the lights will change so that it corresponds to the *state*-value of the new phase and the timer will be reset to the duration-value of the next phase. Since the phases loop, after the last phase is finished the first phase will start again.

How is a traffic light coded as a DNA string?

Now that we know how traffic lights are coded in SUMO, we need to find a way to represent this as a DNA-string. Since we want our GA to be able to manipulate individual lights and the timings of each individual light at an intersection, the DNA-string needs to be able to deal with this. Therefore, a DNA interpretation has been chosen that states for each light at an intersection how many states there are for this light, what its starting state is, and a list of timings so that we know at what moment this light will transition to the next state. We have used a form of value encoding as stated by Malhotra et al. (2011). The value encoding method is less commonly used with GA’s than other methods such as a bit-string variant, but it is the best option given the constraints of working with SUMO. By using value encoding we can represent the states as seen in the previous chapter, as well as the moments when a transition should occur in our DNA-string. Therefore, the DNA-string contains for each light the following genes:

First, we have decided that the DNA-string must contain the number of states within one phase. Since we know that a light will always go from red to yellow to green and back to red, the list of states needs to be a multiple of three. Otherwise, it cannot loop back to the initial state. This way there will not be conflicting states such that a yellow light comes after a red light. We chose to represent this number of states-gene by the number of cycles in the DNA-string. We did this because this makes it easier to apply mutation on these types of genes. Therefore, this number is how many multiplications of three states there are. ‘1’ means that it contains 3 states, and ‘3’ means that this light has 9 states.

Next is the initial state. Because we can infer all states from the initial state, we only need to encode this initial state. This has been done with an integer, where ‘0’ corresponds to green, ‘1’ with yellow and ‘2’ with red. We chose to use an integer instead of the letter it corresponds with because it is easier to apply mutation on these types of genes.

Lastly is a list that contains the timings that initiates a transition to the next state. This list has the largest effect on a traffic simulation. This is because this list decides for each light how long it will stay either red, green, or yellow. If one movement is being benefitted by longer green lights, it will be represented here. The sum of the timings list will be the same for each light at the intersection since the initial states for each light have to line up. This is a constraint within SUMO’s programming, as it would otherwise not be possible to write the program so that SUMO could load it in when simulating.

Since that all these individual components are being selected for each individual light, we need to combine them so that the DNA-string is representative for all lights in the intersection. Because our model contains 14 lights, we have $14 \times 3 = 42$ items in our DNA string. They are written in order, so the final DNA-string looks like the following, followed by a possible interpretation of this:

[number of cycles light 1, initial state light 1,[timings for light 1], number of cycles light 2,(...),[timings for light 14]]
 [1, 2, [25,35,30], 2, 0, [15,10,12,28,10,20], (...), [22,45,23]]

Before we can answer how the core principles of a GA – selection, crossover, and mutation – are implemented, we need to answer our sub-questions. These answers are important because they influence what offspring is generated.

Answering SQ3: How does a fitness measure look like so that it considers waiting times and vehicular uses?

As stated in the introduction, the goal of this research is to find a set of traffic light timings such that car usage will be reduced while traffic flow is not coming to a standstill. The fitness function of the GA should have the same goal: DNA-strings that perform well according to these values are more likely to produce offspring.

So, for the fitness function we are interested in two components: average waiting times and vehicular uses. Luckily, after a simulation in SUMO we can easily obtain these values. After each performed simulation an XML-file that contains information about each agent is outputted. This document contains for each agent its ID as well as its ‘timeloss’.

```
<personinfo id="6" depart="6.00" type="DEFAULT_PEDTYPE">
  <ride waitingTime="0.00" vehicle="6_0" depart="6.00" arrival="39.00" arrivalPos="48.74" duration="33.00" routeLength="346.89" timeLoss="10.30"/>
</personinfo>
```

Information about an agent after a simulation. This agent was assigned ID 6 and had a timeloss of 10,30.

With the ID-value of each agent we can find out what method of travel it has used. When instantiating the agents, we have kept a list that contains for each agent its ID and mode of travel, so whether that agent is a car driver, pedestrian, cyclist, or transit user. The ‘timeloss’ value represents the time an agent has lost compared to if it were able to move at the maximum allowed speed.¹

With this data we were able to determine the percentage of agents that used an eco-friendly alternative for travel, as well as the average timeloss for all agents. These two components are combined in the function that determines the fitness of a DNA-string:

$$F = (\% \text{ of agents using car, bike and walking}) - (\text{average timeloss})$$

Importantly, we can see from this function that the second component is being punished via the negative operator, while the first component is being rewarded. This means that a high average timeloss drags the fitness score down, while a high percentage of eco-friendly usages boosts the fitness score. This means that the fitness function should reduce car-usage without destroying traffic flow.

For the fitness function we have used the percentage of agents that use an eco-friendly alternative. However, we have not yet explored how an agent decides when to use one mode of transportation. This will be explored when we answer the next sub-question:

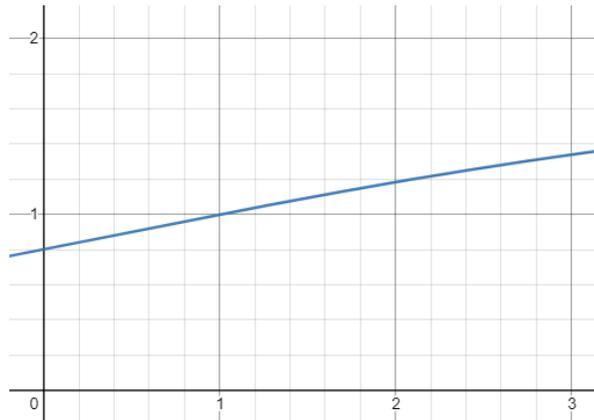
¹ The time lost due to travelling at speed below the maximum speed in that stage. For a <ride> this is the timeLoss of the vehicle during the ride.
 Obtained from SUMO documentation: <https://sumo.dlr.de/docs/Simulation/Output/TripInfo.html>

Answering SQ4: When do drivers move away from driving a car and instead use other modes of transportation?

Each DNA-string is simulated 5 times before its fitness values is calculated. The initial chances of agents choosing to use a particular mode of transportation are equal for all modes. In our simulation agents travel by either foot, bicycle, bus, or car. Therefore, we have chosen the initial probabilities to be 25% for each mode of transportation. After each of these 5 simulations, agents will reevaluate their vehicular uses. This reevaluation is done on population level, so only the probabilities for the whole population change.

We can imagine that if one were to travel from point A to point B one would be more inclined to take the path of least resistance. If one chooses to travel by bicycle but must wait for a long time at an intersection while it appears that cars are finding less resistance, they might decide to go by car next time. This principle has been used in our code.

To calculate the new chances for the population to either drive a car, walk, cycle, or travel by bus, we use the hyperbolic $\tanh\left(\frac{x}{5}\right) + 0.8025$ function. x is defined as the division of the average timeloss for eco-friendly uses over the average timeloss for cars. Timeloss will always be a positive value, since agents cannot travel than their maximum allowed speed, thus x will always be a positive value. Theoretically a division through 0 would be a possible occurrence, but in practice it never happens that none of the agents experience zero timeloss.



The used hyperbolic function. Important is that it crosses at $x=1, y=1$. This means that if the division of the two timelosses is equal to zero, agents will not change their habits.

This function allows us to be lean on small differences between timeloss of the different modes, while being harsh on large differences, without punishing outliers extremely harsh. This makes sense: If one has a lot more resistance in the form of timeloss while driving compared to cycling, they would easily go by bike next time. However, if the difference were small and the timeloss is nearly the same for both modes, one would be more likely to stick to their habits and use the same mode of travel. On the other side if the difference in timeloss were extremely high, say one thousand times, one would probably be as likely to change modes as one with a relatively high amount of timeloss, that is for example 100 times larger.

How are selection, crossover and mutation applied?

As stated in the ‘Method’ chapter, truncation selection has been chosen. This has been implemented by selecting the top 50 percent of DNA-strings according to their fitness scores so that only they are allowed to produce offspring. The rest of the generation will be discarded as they are not performing well enough according to the fitness function.

After selection parents are randomly paired, and single point crossover is applied. We have chosen not to do this for each individual gene of the DNA-string, but for each ‘traffic light’ in the DNA string. This is because the ‘number of cycles’, ‘initial state’ and ‘timings list’ genes are interlinked. This is important for our implementation, because we can imagine that if a crossover happened within such a group a traffic

light could break, as we can see by the following example. If we instead only allowed crossover to occur between traffic lights, this problem is avoided.

Parent A: [2, 2, [10,10,10,10,10,10]], Parent B: [1, 0, [20,20,20]]

Child 1: [2,2,[20,20,20]], Child 2: [1, 0, [10,10,10,10,10,10]]

A situation where regular crossover would cause a problem.

The generated children cannot be created into a traffic light, since for both children the 'number of cycles' value is not the same as the length of the timings list.

Mutation occurs after the selection and crossover processes and happens to the generated children. Each child has a 1% chance of being mutated. This value has been chosen based on the recommendations from Malhotra et al's paper (2011). If a child is being mutated, a random genome out of its DNA-string will be selected, and according if it is a 'number of cycles', 'initial state' or 'timings list' genome, different things occur.

If a 'number of cycles' genome is being mutated, this value will be either upped by one or reduced by one, and the 'timings list' genome will be updated accordingly. If the 'number of cycles' genome is being upped, the values in the timings list will be reduced and 3 new timings will be randomly generated such that the sum of all timings will stay the same.

If instead the 'number of cycles' genome were to be reduced by one, the last 3 timings in the 'timings list' will be removed and their values will be spread over the remaining values.

If an 'initial state' genome is selected for mutation it will be upped by one or reduced by one, so that it becomes one of the other two possible states.

Lastly a 'timings list' genome could be selected for mutation. When this is the case, permutation mutation will occur. This means that two members from the 'timings list' will be swapped.

[1,2,[15,10,5]] -> [2,2,[7,5,2,3,5,8]]

Before and after mutation of the 'number of cycles' genome.

Because the timings list is linked with this value, it will be updated too.

[1,2,[15,10,5]] -> [1,1,[15,10,5]]

Before and after mutations of the 'initial state' genome.

[1,2,[15,10,5]] -> [1,2,[5,10,15]]

Before and after mutations of the 'timings list' genome.

Two timings are being swapped places. This is a form of permutation mutation.

4. Results

In this chapter we will present the results that were generated by the model. We have used a population size of 28 DNA-strings. This value was chosen because of the recommendations in the paper of Malhotra et al. (2011). We have simulated the model for 30 generations and with this data we will answer our research question:

How could traffic light timings be used to reduce car usage?

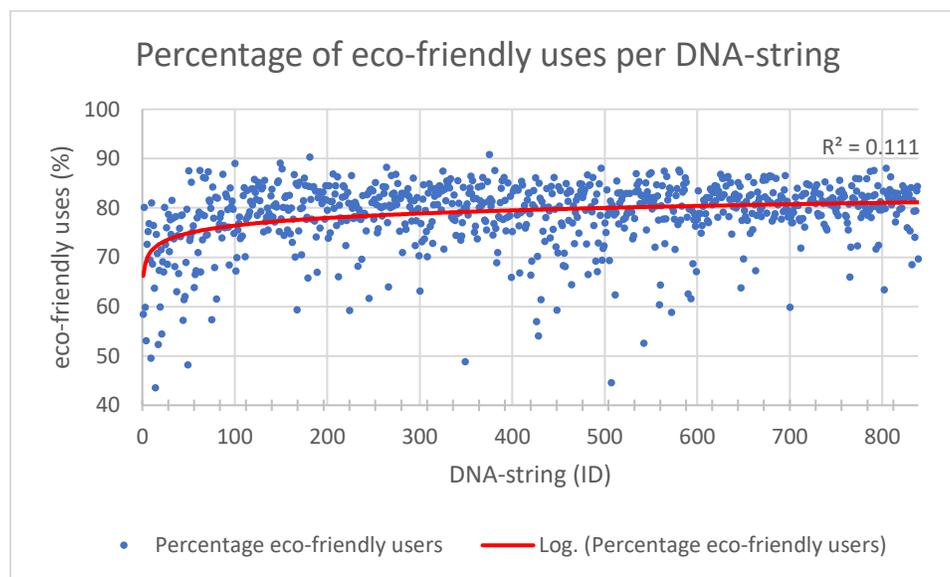
We have set two sub-questions that should lead us to an answer to this question, and we will start by examining sub-question 1. Here we asked ourselves the following question:

Answering SQ1: Do agents change their behavior such that they use other modes of mobility?

To answer this question, we select for each simulated DNA-string what forms of mobilities agents used to move around. We only distinguish between ‘eco-friendly’ uses – that is transportation by either foot, bicycle, or bus – and transportation by car. We group these three modes because the model has promoting eco-friendly uses as its goal and does not prefer one eco-friendly mode of transportation over another eco-friendly mode. It only wants to get as many people use one of these three modes of transportation and as least people choosing to drive a car, without causing a standstill for cars.

The percentage of agents that used an eco-friendly mode of traffic is plotted in Graph 1. On the x-axis we have set out all DNA-strings, in order of simulation. The DNA-string with ID 0 was simulated first, and DNA-string 839 was the last simulated DNA-string. Because the generation size was 28, DNA-strings 0 to 27 were the first generation, DNA-strings 28 to 55 were the second generation, and so on.

A trendline has been plotted that has an R-squared value of 0.111. According to Cohen (1992) this is classified as a low correlation.



Graph 1: The percentage of eco-friendly uses per intersection. As more natural selection occurs, this value trends upwards to around 80.4%.

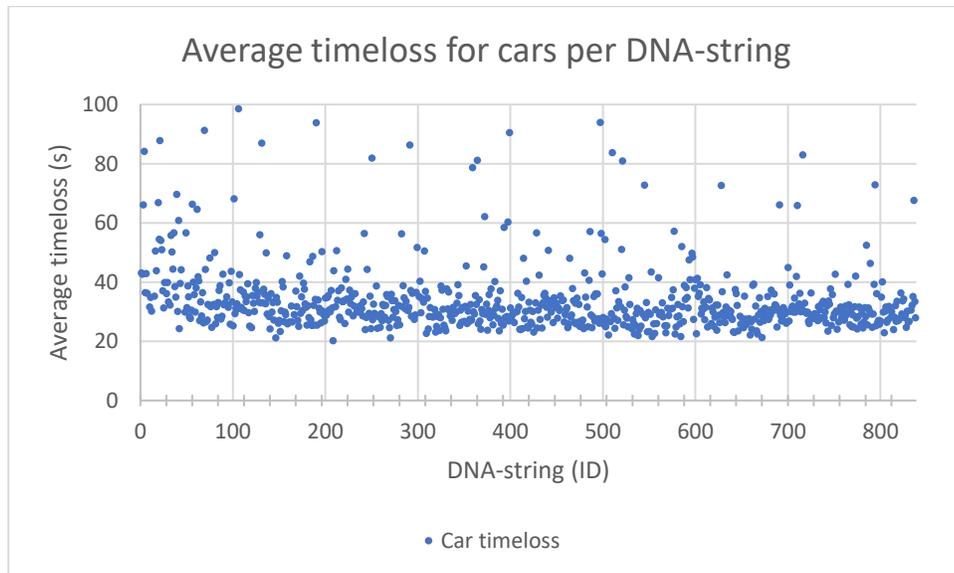
The average percentage of eco-friendly users in the first generation (DNA-strings 0 to 27) was 67.6 percent, while in the last generation (DNA-strings 812 to 839) this value rose with 12.8 percentage point to 80.4 percent.

We see that as more natural selection occurs, more agents changed their behavior and preferred to travel by an eco-friendly alternative. This means that the GA was able to select traffic lights that sway agents more away from driving a car, as it was designed to. Next, we will examine sub-question 2:

Answering SQ2: Do successful traffic lights punish car drivers with longer waiting times?

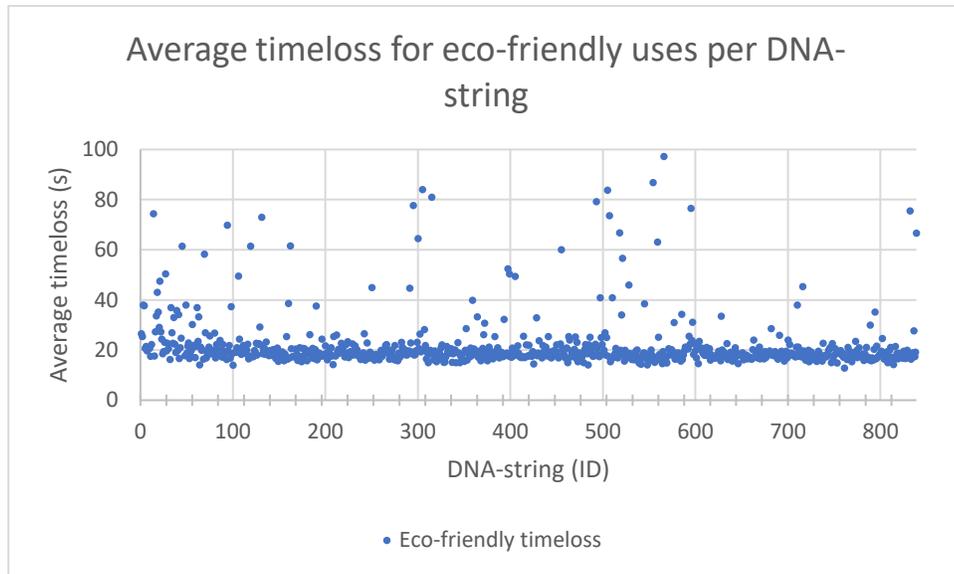
To measure the time agents spent waiting, we will use the timeloss measure that was introduced in the fitness function. This value is useful because it only takes involuntary waiting into account. By this waiting for a bus at the bus stop is not counted, because it is a voluntary action: if one were to travel by bus, they would know that waiting for a bus is part of that action.

From our data based on the timeloss measure we see that cars are being punished. In Graph 2 we have set out the average timeloss experienced by car drivers per DNA-string. Although there is a lot of spread, we see that the majority of all timeloss is in the 25-35 seconds range. The average timeloss in the first generation is 72.5 seconds, but it has been reduced to 37.7 seconds through natural selection.



Graph 2: Average timeloss for cars. Note that the y-axis has been cut-off at 100 seconds to remove large outliers.

For eco-friendly uses we see that these values are lower. In Graph 3 we have plotted the average timeloss experienced by agents using eco-friendly modes of transportation. For this group we again see a lot of spread in the data, but the majority of timeloss data is in the range of 15-25 seconds. In the initial generation the average timeloss was 39.7 seconds, and in the last generation been reduced to an average of 22.0 seconds. Thus, timeloss for eco-friendly uses shrunk with 15.7 seconds and timeloss for cars shrunk by 34.8 seconds, but car users lost on average 15.7 seconds more compared to their eco-friendly counterparts.



Graph 3: Average timeloss for eco-friendly uses. The y-axis has been cut-off at 100 seconds to remove outliers.

From this we can conclude that cars were indeed punished in the form of higher timeloss values. However, it is important to analyze where this timeloss originates from. For each DNA-string we can compare the average percentage of time a traffic light was in a ‘green’ state. For traffic lights we cannot differentiate between car drivers and users of eco-friendly alternatives. It is however possible to compare the percentage of time green lights were shown to motorized traffic (cars and busses) and the percentage of time green lights were shown to non-motorized traffic (pedestrians and cyclists).

We found that in generation 30, where the percentage of agents choosing to use the most eco-friendly was the highest, that the percentages of green light converged to 28.1% and 37.2% respectively. This shows us that punishing car drivers in the form of shorter timings for ‘green’ states for motorized traffic compared to non-motorized traffic successfully causes agents to change their behavior.

Lastly, we will compare the – according to percentage of eco-friendly uses – worst performing DNA-string from generation 1 to the best performing DNA-string from generation 30. Comparing these DNA-strings allows us to see what makes a successful set of traffic light timings.

<p>[1, 2, <u>[34, 27, 29]</u>, 4, 1, [1, 7, 2, 11, 1, 11, 13, 15, 5, 1, 1, 22], 1, 1, [37, 25, 28], 4, 2, [14, 8, 7, 8, 2, 11, 10, 1, 1, 1, 1, 26], <u>2, 2, [10, 11, 21, 11, 26, 11]</u>, 1, 1, [29, 38, 23], 4, 2, [8, 6, 18, 16, 1, 4, 1, 1, 9, 1, 1, 24], <u>3, 0, [1, 7, 15, 2, 10, 18, 12, 1, 24]</u>, 1, 1, [27, 31, 32], 2, 1, [12, 17, 21, 1, 13, 26], 1, 0, [39, 15, 36], <u>4, 0, [3, 5, 11, 8, 6, 13, 5, 15, 1, 1, 1, 21]</u>, 4, 2, [7, 6, 3, 9, 1, 16, 13, 11, 1, 1, 1, 21], 3, 2, [2, 13, 11, 10, 14, 6, 2, 1, 31]]</p>	<p>[3, 1, [8, 11, 21, 12, 5, 7, 6, 15, 5], 4, 0, [6, 7, 2, 2, 1, 8, 3, 1, 15, 14, 14, 17], 4, 2, [5, 2, 5, 1, 8, 3, 9, 1, 9, 12, 22, 13], 3, 1, [4, 20, 12, 8, 10, 3, 7, 2, 24], <u>4, 1, [7, 12, 2, 8, 2, 2, 11, 19, 1, 1, 1, 24]</u>, 3, 0, [7, 12, 14, 10, 11, 1, 6, 1, 28], 3, 0, [9, 9, 7, 9, 9, 8, 3, 1, 35], <u>2, 1, [12, 18, 12, 16, 15, 17]</u>, 2, 1, [13, 15, 14, 24, 12, 12], 2, 1, [12, 22, 10, 12, 23, 11], 4, 1, [3, 9, 7, 4, 10, 11, 12, 7, 4, 1, 1, 21], <u>4, 1, [11, 9, 1, 13, 2, 4, 2, 8, 8, 1, 1, 30]</u>, 4, 0, [8, 11, 2, 7, 7, 1, 1, 9, 15, 5, 1, 23], 4, 1, [9, 9, 8, 4, 18, 10, 5, 7, 1, 1, 1, 17]]</p>
---	--

The worst performing DNA-string from the first generation (left) and the best performing DNA-string from the last generation (right). Underlined genes are non-motorized traffic lights. **Bold** timings show ‘green’ timings.

We see that in the worst performing randomly generated DNA-string compared to the best of generation 30, for lights dedicated to motorized and non-motorized traffic shorter cycles are more prominent. In the worst performing DNA-string there are 5 lights that contain just 1 cycle of red-green-orange, and that the average cycle length is 2.5. In the best performing DNA-string of the last generation we see that lights containing 1 cycle have been selected out, and that the average cycle length was 3.3. Shorter but more frequent lights seem to perform better. This is likely because it wastes less time on green when a queue is cleared and is therefore more efficient. It allows shorter but more frequent bursts of green lights and transitions to yellow and red states shorter after the queue has been cleared.

Another change is in the percentage of time lights are showing green. The worst-performing DNA-string shows green lights to motorized traffic 27.6 percent of the time and to non-motorized traffic 26.6 percent of the time. Meanwhile the best performing DNA-string is showing green for lanes carrying motorized traffic 28.1 percent of the time and for lanes carrying non-motorized traffic 37.2 percent of the time. This means that green light timings for non-motorized traffic improved with 0.5 percentage point, while green light timings for non-motorized traffic improved with 10.6 percentage point. Showing green lights for longer periods to eco-friendly modes as walking and cycling makes more people to use those modes.

5. Conclusion & Discussion

The model shows us that changes in traffic light timings can be altered in such a way that users are changing their behavior. We have seen that traffic lights with longer durations of red lights for motorized traffic compared to lights that contained only non-motorized mobilities and thereby punishing drivers works, with agents changing their behavior. Cycles with more and shorter state durations proved to be more successful in doing this than cycles with less but longer state durations. Showing green lights for longer periods of time to non-motorized modes of transportation swayed agents to use eco-friendly uses. A genetic algorithm proved a useful tool in exploring the set of randomly generated traffic lights and was able to improve the percentage of agents using eco-friendly transportation modes with 12.8 percentage points.

It is important to state that these results are based on certain assumptions. These assumptions were made in answering sub-question 4, when modelling when people change their modes of travel based on experienced timeloss. Further research should therefore focus on answering this in further detail, basing their values on human observations. This way the model would be more accurate in modelling when and how humans behave and what the effects of changes in traffic light timings on their behavior are.

Also, the model did not consider public safety. Although safety was not part of the scope of this study, it is important when implementing traffic light in the real world. For the SUMO software this is not a problem, but in real-world intersections that contain movements that cross on each other on green lights are a hazard for collisions. If an intersection were unsafe for cyclists and pedestrians, these groups would be more inclined to just go by car, given the smaller chance of injuries. The GA could be reworked so that the number of crossing movements would be punished by the fitness function, with the hope that intersections that contain a lot of crossing movements were not able to generate offspring.

Another part where the algorithm could be improved is the implementation of yellow lights. In this model, they are initiated in the same random way as red and green lights. In real-world intersections we see that yellow light timings are set based on speed restrictions, such that drivers have enough time to slow down enough when the light turns from green to red. The result of this is that in our model drivers wait for yellow lights in the same way they do for red lights.

6. References

- Beyzaltar, M. A., Karacal, M., & Yetkiner, H. (2014, May 1). *Granger-causality between transportation and GDP: A panel data approach*. ScienceDirect. <https://linkinghub.elsevier.com/retrieve/pii/S0965856414000524>
- CBS. (2018, November 8). *More vehicle traffic than ever in the Netherlands*. <https://www.cbs.nl/en-gb/news/2018/45/more-vehicle-traffic-than-ever-in-the-netherlands>
- CBS. (2019). *Car ownership - The Netherlands on the European scale*. The Netherlands on the European scale | 2019. <https://longreads.cbs.nl/european-scale-2019/car-ownership/>
- CBS. (2021). *Emissies van broeikasgassen berekend volgens IPCC-voorschriften*. <https://opendata.cbs.nl/statline/#/CBS/nl/dataset/70946NED/table?dl=4191C>
- Cohen, J. (2013). *Statistical Power Analysis for the Behavioral Sciences*. Taylor & Francis.
- Crow, J. F., & Kimura, M. (1979). Efficiency of truncation selection. *Proceedings of the National Academy of Sciences*, 76(1), 396–399. <https://doi.org/10.1073/pnas.76.1.396>
- Energy Cities. (2019, June 19). “*Superblocks*” free up to 92% of public space in Barcelona! <https://energy-cities.eu/best-practice/superblocks-free-up-to-92-of-public-space-in-barcelona/>
- France24. (2020, September 16). *Paris to keep new cycling paths beyond pandemic*. France 24. <https://www.france24.com/en/20200916-paris-to-keep-new-cycling-paths-beyond-pandemic>
- Goodyear, S. (2014, November 20). *The Swedish Approach to Road Safety: “The Accident Is Not the Major Problem.”* Bloomberg CityLab. <https://bloom.bg/3wBwXa9>
- Ile-de-France. (2020). *RER V - le réseau vélo d’Île-de-France*. RER V. <https://rerv.fr/>
- Jevons, W. S. (1865). *The Coal Question: An Inquiry Concerning the Progress of the Nation, and the Probable Exhaustion of Our Coal-Mines*. <https://books.google.nl/books?id=gAAKAAAIAAJ&vq=editions%3AAAotKDT6KKcC&hl=nl&pg=PR3#v=onepage&q=editions:AAotKDT6KKcC&f=false>
- Kwantes, C. (2019, October 7). *Een rijdende auto neemt meer plek in dan je waarschijnlijk denkt*. EenVandaag. <https://eenvandaag.avrotros.nl/item/een-rijdende-auto-neemt-meer-plek-in-dan-je-waarschijnlijk-denkt/>
- Malhotra, R., Singh, N., & Singh, Y. (2011). Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. *Computer and Information Science*, 4(2). <https://doi.org/10.5539/cis.v4n2p39>
- Ministerie van Infrastructuur en Waterstaat. (2020, May 8). *Mobiliteitsbeeld 2019*. Kennisinstituut voor Mobiliteitsbeleid. <https://www.kimnet.nl/mobiliteitsbeeld/publicaties/rapporten/2019/11/12/mobiliteitsbeeld-2019-vooral-het-gebruik-van-de-trein-neemt-toe>
- Mobiliteits Zentral. (2020, March 3). *Free transport in Luxembourg*. <https://www.mobiliteit.lu/en/tickets/free-transport/>

NYC DOT. (n.d.). *Past Bicycle Projects*. Retrieved May 26, 2021, from <https://www1.nyc.gov/html/dot/html/bicyclists/past-bike-projects.shtml>

Obordo, R. (2020, February 3). “*The streets are more alive*”: *Ghent readers on a car-free city centre*. The Guardian. <https://www.theguardian.com/environment/2020/jan/20/the-streets-are-more-alive-ghent-readers-on-a-car-free-city-centre>

Oltermann, P. (2020, February 3). *Vienna’s euro-a-day public transport model could waltz into Berlin*. The Guardian. <https://www.theguardian.com/world/2019/jul/09/vienna-euro-a-day-public-transport-berlin-365-annual-ticket>

Parkopedia. (2020). *2019 Global Parking Index Report*. <https://business.parkopedia.com/2019-global-parking-index>

Sisson, P. (2020, November 18). *Ban cars: Why cities are embracing the call for car-free streets*. City Monitor. <https://citymonitor.ai/transport/ban-cars-why-cities-are-embracing-the-call-for-car-free-streets>

Teo, K., Kow, W., & Chin, Y. (2010). Optimization of Traffic Flow within an Urban Traffic Light Intersection with Genetic Algorithm. *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*. Published. <https://doi.org/10.1109/cimsim.2010.95>

Turky, A. M., Ahmad, M. S., & Yussof, M. Z. M. (2009). The Use of Genetic Algorithm for Traffic Light and Pedestrian Crossing Control. *IJCSNS International Journal of Computer Science and Network Security*, 2009(9.2), 88–96.

Wikipedia. (n.d.). *OnePointCrossover* [Illustration]. Crossover (Genetic Algorithm). <https://bit.ly/3x8F1Q7>