TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

Siim Salin 176981IAPM

SMART AND SUSTAINABLE TRAFFIC SIGNAL MANAGEMENT USING IMPROVED DYNAMIC SCHEDULING ALGORITHM IN THE CITY OF TALLINN

Master's thesis

Supervisors:	Sadok Ben Yahia
	PhD
	Dago Antov
	PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Salin 176981IAPM

TARK JA JÄTKUSUUTLIK VALGUSFOORIDE JUHTIMINE KASUTADES TÄIUSTATUD DÜNAAMILIST VALGUSFOORI JUHTIMISE ALGORITMI TALLINNAS

Magistritöö

Juhendajad:	Sadok Ben Yahia
	PhD
	Dago Antov
	PhD

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Siim Salin
May 13, 2020

Abstract

The main subject for this thesis is to implement and evaluate a dynamic traffic signal scheduling algorithm and evaluate its performance on three most congested intersections in Tallinn that meet the requirement of the algorithm – Tammsaare-Sõpruse, Tammsaare-Mustamäe, and Endla-Paldiski-Mustamäe intersections. The introduced algorithm reduces vehicles' delay time and fuel consumption while prioritizing more public transportation using a passive priority system. The algorithm gathers real-time traffic characteristics of each competing traffic flow to choose the next signal phase and the length of the green phase. The optimal protection phase is chosen to schedule the next signal phase.

The evaluation of the algorithm is made using a realistic simulation environment for Tallinn's intersections. For each intersection, the simulation environment was prepared with realistic traffic network and vehicle demands. Traffic signal schedules were measured for given intersections and implemented in a simulation environment for baseline evaluation.

Introduced algorithm were integrated with simulation environment. Simulation results with the baseline traffic signal schedule for peak and off-peak hours were compared to introduced algorithm results. The introduced algorithm showed excellent results for peak and off-peak hours. The measures compared were vehicles' trip duration, delay time and fuel consumption.

The thesis is in English and is 42 pages long, including 6 chapters, 25 figures, and 11 tables.

Annotatsioon

Töö põhieesmärgiks oli implementeerida ning hinnata dünaamilist valgusfoori signaali algoritmi kolmel ristmikul, millel võib tekkida ummikuid tipptunnil ning mis vastavad tutvustatud algoritmi eeldustele. Nendeks ristmikuteks on Tammsaare-Sõpruse, Tammsaare-Mustamäe ning Endla-Paldiski-Mustamäe. Tutvustatud algoritm eelistab liiklusvooge, kus liigub rohkem ühistranspordisõidukeid, samal ajal vähendades tavasõidukite ooteaega ning kütusekulu. Ühistranspordi prioritiseerimine on ehitatud kasutades passiivset eelistussüsteemi. Pakutud algoritm kogub reaalajas infot ristmiku liiklusvoogude kohta ning selle põhjal arvutab välja optimaalse järgmise signaali takti ning selle pikkuse. Takti planeerimiseks leitakse optimaalne kaitsetakt.

Tutvustatud algoritmi hindamine teostatakse simulatsioonikeskkonnas, milles on implementeeritud kolm eelnevalt mainitud ristmikku. Iga ristmiku jaoks on kasutatud realistlikku liiklusvõrgustikku. Võrgustikku on lisatud mootorsõidukid, kasutades reaalelulisi andmeid. Praeguse olukorra ehk algtaseme defineerimiseks mõõdeti kõigi kolme ristmike valgusfooride signaalide tsüklid, mille järgi ehitati simlatsioonikeskkonda valgusfoori tsüklid.

Tutvustatud algoritm integreeriti simulatsioonikeskkonnaga. Simulatsioonitulemused mõõdeti nii algtaseme kui ka tutvustatud algoritmi kohta. Tutvustatud algoritm näitas suurepäraseid tulemusi nii tipptunni kui ka tipptunni välise aja kohta. Mõõdikuteks kasutati sõidukite reisikiirust, ooteaega ning kütusekulu.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 42 leheküljel, 6 peatükki, 25 joonist, 11 tabelit.

List of abbreviations and terms

ITLC	Intelligent Traffic Light Controlling
VANET	Vehicular Ad-hoc Network
ETLSA	Enhanced Traffic Light Scheduling Algorithm
PETSSA	Priority-driven Enhanced Traffic Signal Scheduling Algorithm
SUMO	Simulation of Urban Mobility
SUMO-GUI	Simulation of Urban Mobility Graphical User Interface
ATL	Arterial Traffic Light
CCTV	Closed-Circuit Television
DCNN	Deep Convolutional Neural Network
PCA	Principal Component Analysis
DCSP	Distributed Constraint Satisfaction Problem
OSM	OpenStreetMap
POI	Points of Interest
XML	Extensible Markup Language
OD	Origin-Destination
TLS	Traffic Light Schedule
TraCi	Traffic Control Interface
TCP	Transmission Control Protocol

Table of contents

List of Figures

List of Tables

1 Introduction

Traffic signals are placed throughout the road networks to control competing traffic flows at intersections. These traffic signals split time-space between conflicting flows and enhance safety of vehicles while crossing intersections by scheduling conflicting traffic flows. However, traffic signals may decrease vehicles' efficiency in traffic networks. This decrease happens because vehicles must wait for the green phase of the traffic signal to pass through the intersection. It is of paramount replace to use the optimal signal schedule. The optimal green phase can reduce waiting time as well as fuel consumption [24, 26]. One of the most potent ways to lower pollution is by using public transportation. Prioritizing more public transportation could make it more competitive. Both efficient green phase and prioritized public transportation throughput could be improved using real-time dynamic traffic signal algorithm.

Intelligent Traffic Light Controlling (ITLC) [26] uses Vehicular Ad-hoc Network (VANET) technology to gather real-time traffic characteristics of each competing traffic flow at isolated traffic signal intersections. The ITLC algorithm has demonstrated excellent performance in terms of decreasing the waiting delay time of traveling vehicles and increasing the throughput of the signalized intersections [26]. Although it does not measure the performance of public transportation, transit signal priority could be improved using a passive priority system with ITLC. Furthermore, ITLC would have some weaknesses in Tallinn - ITLC or the further developed ETLSA [5] are designed to use cooperative communication such as VANET to get input parameters for the algorithm, and the performance have been only designed for 4-leg traffic intersection.

In this thesis, the introduced algorithm is a further development of ITLC. The algorithm aims to design a Priority-driven Traffic Signal Scheduling Algorithm (PETSSA) that considers passive priority of public transportation over the intersection flows. This algorithm has three main stages. The first stage defines the phases based on the intersection type. In the second stage, the algorithm is enhanced to consider public transportation efficiently. The enhanced algorithm changes the maximum green phase time of traffic flow efficiently to prioritize flows with high public transportation traffic. Thus, each public transportation vehicle has more time to pass the congested intersection. The third stage considers designing a dynamic traffic signal scheduling algorithm that sets the phases of each traffic signal cycle based on the traffic distribution of competing traffic flows. Real-time traffic characteristics of each traffic flow are measured using camera output data and speed sensors among intersection flows. Then, the highest-density competing traffic flow is scheduled to pass the signalized intersection first, coupled with one or more non-conflictive flow.

Furthermore, the simulation environment is developed, and numerous sets of experiments have been used to evaluate the performance of the introduced algorithm using the three most congested intersections in Tallinn - Tammsaare-Sõpruse, Tammsaare-Mustamäe, and Endla-Paldiski-Mustamäe. Results are compared to the baseline simulation results, which are run on real-life signal schedules.

The main goal is to implement a dynamic traffic signal scheduling algorithm and evaluate its performance on the three most congested intersections in Tallinn. The algorithm relies on the use of camera video data, passive public transportation prioritization, and is integrated with the simulation environment SUMO. Thus the performance is validated using real-life scenarios.

2 Background

2.1 Intersections and traffic signals schedules in Tallinn

Traffic signal control in traffic networks is an essential practical problem due to substantially increasing delay and fuel consumption caused by more vehicles in traffic. It is well known that an improved traffic signal control may offer great potential for reducing congestion on streets, and that signal schedules that adjust their settings to fit current traffic conditions may be the best choice.

The classical signal schedule assumes a cyclic operation of traffic lights, where each signal moves through a sequence of phases with calculated split parameters in a cycle that is offset from those of its neighbors (D. Antov, personal communication, February, 2020). In Tallinn, the signal cycle is from 72 to 90 seconds (D. Antov, personal communication, February, 2020). Using dynamic traffic signal control nullifies the meaning of cyclic operation as each intersection operates on its own. In this thesis, the introduced algorithm cycle length is not limited. The best performing configuration is determined and used.

Furthermore, some of the Tallinn's intersections are already equipped with cameras (D. Antov, personal communication, February, 2020). Tallinn University of Technology has installed speed sensors to various locations in Tallinn (J. Kaugerand, personal communication, February, 2020). Cameras and sensors form an excellent environment to parse dynamic traffic data in real-time. It is possible to use speed sensors further away from the intersections to measure average traffic speed, as well as cameras to assess the density of vehicles near intersections. These measures are necessary to introduce a dynamic algorithm.

Several intersections need a better traffic signal schedule and fit well for the needs of the introduced algorithm in this thesis. The chosen intersections must have following characteristics (D. Antov, personal communication, February, 2020):

- There should be a remarkable number of public transportation vehicles to achieve a priority-based effect;
- There should be congested situations, meaning that average delay is high;

- There should be non-existing public transportation priority introduced today.

Considering the mentioned characteristics, three intersections are used for introduced algorithm performance evaluation as they all meet the requirements (D. Antov, personal communication, February, 2020):

- Tammsaare-Sõpruse;
- Tammsaare-Mustamäe;
- Endla-Paldiski-Mustamäe.

All of the chosen intersections do not have an existing public transportation priority system and also have highly congested situations on peak hours. In general, Tallinn has many congested intersections with the requirements for a dynamic traffic signal schedule.

2.2 Dynamic traffic signal scheduling

One way to enhance traffic signal scheduling is using dynamic traffic signal schedules. Many studies aim to enhance traffic efficiency over the network and decrease the delay time of vehicles. Mostly, dynamic real-time traffic signal schedules can decrease delay time and save fuel. Vehicular ad-hoc networks (VANET) and sensor networks have been utilized to gather and aggregate the real-time traffic information of competing traffic flows at each intersection [7, 11, 17, 20, 21, 25–27]. Due to the development of artificial intelligence, complex, intelligent schedules have been used to schedule the phases of the traffic signals, e.g., Fuzzy Logic [3], Genetic algorithm [8] and Oldest Job First algorithm [21]. More interesting is the reduction of fuel consumption and gas emission while using dynamic traffic signals systems. There have been studies using VANET technology to study the effect of traffic signal controlling systems on fuel consumption [24] and gas emissions [10]. As can be seen, intelligent traffic light systems can be classified into three main categories, based on the technology utilized: mathematical modeling and machine learning; artificial intelligence technologies; and VANET communication technology.

Intelligent Traffic Light Controlling (ITLC) [26] is a fundamental algorithm in this thesis. It uses VANET technology to gather real-time traffic characteristics of each competing traffic flow at isolated traffic signal road intersections [26]. These traffic characteristics are considered while setting the sequence of phases and the time of each phase in the traffic signal timing cycle [26]. In ITLC, the densest traffic flow is scheduled to cross the signalized intersection first [26]. Moreover, the time of each phase is set based on the location and speed of the last vehicle that is expected to cross the signalized intersection during the scheduled phase [26]. The ITLC algorithm has demonstrated an excellent performance in terms of decreasing the waiting delay time of traveling vehicles and increasing the throughput of the signalized intersections [26]. The best green signal length calculation of ITLC is used in this thesis for the introduced algorithm. The majority of the previous ITLC algorithms have been designed for isolated traffic signal scenarios (i.e., a traffic signal that schedules the traffic timing cycles without considering nearby signalized intersections). The communications of VANETs have been used to gather or deliver the traffic characteristics of all competing flows to the located traffic signal [26]. ITLC is used with Arterial Traffic Light (ATL) controlling algorithm [26].

Furthermore, the Enhanced Traffic Light Scheduling Algorithm (ETLSA) [5] is a further development of ITLC. It aims to design a dynamic traffic signal scheduling algorithm that considers the presence of one or more emergency vehicles over the road network [5]. The study proved that although the proposed enhanced algorithm decreased the throughput of the signalized intersection, the waiting delay time of emergency vehicles is decreased compared to previous scheduling algorithms [5].

As can be seen, several approaches have been applied to solve the traffic signal scheduling problem considering real-time traffic characteristics, the shapes of the signalized intersections, accidents or roadblock scenarios, emergency vehicles. Mentioned studies did not pay heed prioritizing public transportation. Furthermore, most of these algorithms are using VANET communication and are tested in a simulation environment that supports VANET. In Tallinn, VANET communication is not implemented yet. An alternative would be to analyze camera video and use speed sensors. During the last few decades, significant research efforts have been devoted to using closed-circuit television (CCTV) cameras to determine real-time traffic parameters such as volume, density, and speed [28]. These methods can be divided into three categories: detection-based methods; motion-based methods; and holistic approaches.

Detection-based methods use individual video frames to identify and localize vehicles and thereby perform counting tasks such as neural network methods [19], Kalman filter-based background estimation [4]. Achievements in deep learning methods have led to several methods being used for traffic counting tasks, e.g., DCNNs [1]. Motion-based methods have been used microscopic parameters in a video sequence [2]. This method tends to fail at a low frame rate due to a lack of motion information. Holistic approaches analyze the whole image to estimate the overall traffic state, e.g., spatiotemporal Gabor filters [12] that classifies traffic videos into different congestion types.

Overall, recent studies have shown signs that camera video detection can be used to determine real-time traffic parameters such as volume, density, and speed. A study [22] presents an adaptive traffic light control system using a camera as an input sensor that providing real-time traffic data. Principal Component Analysis (PCA) is used to analyze and to classify moving objects for detecting vehicles and non-vehicles [22]. Distributed Constraint Satisfaction Problem (DCSP) method determines the duration of each traffic light, based on the counted number of vehicles at each lane [22].

It is possible to extract input for dynamic algorithms without VANET, combining processed camera video output with speed sensors [15], which means that each intersection is a separate piece of the traffic network. In this thesis, it is expected that data from camera detection and speed sensors are reliable, and extracting the data from a camera to the desired form is out of the scope of this thesis. In this case, the dynamic scheduling algorithm can work efficiently at Tallinn's intersections and enhance traffic performance. However, combining dynamic scheduling algorithms with public transportation priority is a useful improvement for better signal schedules for creating reliable and efficient public transportation schedules.

2.3 Public transportation priority

Combining a dynamic traffic signal algorithm with prioritized public transportation signals could be a pivotal improvement to make it more desirable while reducing regular vehicles' fuel consumption and delay time. Existing transit signal priority strategies fall into three major categories [23]: passive, active, and adaptive.

Passive priority strategies are developed offline based on historical data and do not require any traffic detection system. The main changes are in the signal settings such as green times, offsets, and cycle lengths [23]. One of the prioritizing methods is to adjust the offset to consider slower buses. On the other hand, the addition of a green phase is a method to raise the probability that transit vehicles will be arriving during the green phase [23]. Reduction in the cycle length is another strategy commonly used as it increases the turnover of the phases and, as a result, decreases the delays for all vehicles [23]. Passive priority systems do not need expensive equipment to implement, as dynamic traffic measuring is not needed. However, their success depends on having traffic volumes with low variability [23].

On the other hand, active priority systems measure traffic in real-time and respond accordingly, therefore making it more effective than passive priority system [23]. VANET is one technology that could be useful to obtain real-time information about vehicles. Active priority systems include extending the green phase until transit vehicles clear the intersection or advancing the start of the green phase for the flow serving a transit vehicle [23]. Furthermore, a new phase can be inserted to serve the transit vehicle at the moment it arrives at the intersection or change the phase cycle as the transit vehicles would be given green phase as soon as possible [23]. As active priority systems need detection or vehicle communication systems, it is more expensive to install and implement. Active priority systems tend to be more effective as they can adapt to the variability of the traffic.

Furthermore, there have been researches using adaptive transit signal priority. A study optimized adaptive transit signal priority using a parallel genetic algorithm [13]. The general approach in this study for considering the impact of the higher occupancy of transit vehicles in adaptive transit signal priority optimization is to assign a higher weight to transit vehicles [13]. In addition, the study found out that macroscopic models are not suitable for modeling adaptive signal control since they do not consider individual vehicle arrivals, which are necessary input to adaptive traffic signals. Therefore, the study used a microscopic simulation [13].

In this thesis, both significant categories are used. Passive priority strategy and the adaptive phase length are used to prioritize transit flows. A combination means that the maximum green phase length is extended according to the priority level of the flow if the overall traffic flow is congested and demanding. Furthermore, the priority levels are assigned based on historical public transportation data, and the overall traffic signal schedule is determined by sensing technologies considering all vehicles, no matter their type. This approach would reduce the delay time of all vehicles, including transit vehicles. Microscopic simulation is used, to evaluate the performance of introduced signal algorithm, as all vehicles are measured dynamically around intersections with a passive priority strategy to extend the green phase. Each intersection is tested on its own, using real-life scenario simulation.

2.4 Real-life scenario simulation

The real-life scenario is vital to evaluate the performance of the introduced traffic signal algorithm. Accurate knowledge of traffic conditions and dynamics is necessary to implement traffic management simulation. Fortunately, traffic simulation frameworks provide helpful tools to evaluate traffic signal scheduling strategies and their impacts [18]. The traffic simulation framework tools can mainly be broken into four different groups [16]:

1. Macroscopic: average vehicle dynamics like traffic density are simulated;
2. Microscopic: each vehicle and its dynamics are modeled individually;
3. Mesoscopic: a mixture of a macroscopic and microscopic model;
4. Submicroscopic: each vehicle and also functions inside the vehicle are explicitly simulated e.g., gear shift.

The detailed simulation of microscopic models is precise, especially when emissions or individual routes should be simulated [18]. Therefore, this thesis focuses on the microscopic traffic simulation. There are several microscopic simulation tools available for simulating real-life scenarios. Simulation PTV Vissim is well known and used [9], while the activity-based traffic simulation MATSim [1] is open source and freely available [14]. Furthermore, SUMO [2] is freely available and

[1] `https://www.matsim.org/`
[2] `https://sumo.dlr.de/docs/`

published under the Eclipse Public License V2. A lot of model extensions, simulation enhancements, and improvements have been made to SUMO to support generating needed elements for simulation. In general, several elements are needed to simulate traffic. The most important ones are the following [18]:

1. Network data (e.g., roads and footpaths);
2. Additional traffic infrastructure (e.g., traffic lights);
3. Traffic demand.

It is often a time-consuming process to prepare a simulation scenario based on real-world data. SUMO provides a large package of applications to help with this task [18]. Therefore, SUMO is chosen as a simulation environment for this thesis.

3 Priority-driven Enhanced Traffic Signal Scheduling Algorithm - PETSSA

In this chapter, Priority-driven Enhanced Traffic Signal Scheduling Algorithm (PETSSA) is introduced, which is a dynamic traffic signal scheduling algorithm that considers real-time traffic characteristics of each competing traffic flow at road intersections. To do so, we rely on camera input and speed sensors while giving more priority to the flows with statistically more public transportation demand. Similarly to ITLC [26] and ETLSA [5], traffic characteristics are considered while setting the sequence of phases and the time of each phase in the traffic signal timing cycle. Traffic signal maximum green phase time is determined using a passive priority strategy based on historical public transportation data. The number and sequence of phases would be set differently from one cycle to another. Similarly, the scheduled time of the configured phases would be set differently during each cycle.

The design of PETSSA has been introduced in four main phases. The first phase is to determine the non-conflictive flows to create signal phases (Section 3.1). Non-conflictive flows are competing flows that can be scheduled simultaneously. Each phase change needs an optimal protection phase to change phases in a signal cycle. Flows and protection phases are described individually for each intersection. The second phase of PETSSA designs a passive priority strategy using historical public transportation data to set priority levels for each flow of intersection (Section 3.2). The priority level changes the maximum green phase (*MAX-GREEN*) length for each traffic flow. The third phase of PETSSA designs a dynamic traffic signal scheduling algorithm that sets the phases of each traffic signal cycle based on the traffic distribution of competing traffic flows (Section 3.3). The real-time traffic characteristics of each traffic flow are evaluated using camera data around the intersections and speed sensors near the intersections. The fourth phase is a scheduling algorithm that schedules the highest-density competing traffic flow to pass the signalized intersection first, coupled with non-conflictive flows (Section 3.5).

3.1 Traffic phases for different intersections

Traffic signal schedules should be set to allow all non-conflictive competing traffic flows to cross the intersection safely. Therefore at each phase, multiple traffic flows that are not in conflict can be scheduled concurrently. Non-conflictive flows create phases that are used as a sequence in the traffic signal cycles. Considering the typical 4-leg signalized intersection scenario in Tallinn, each phase presents a pair of synchronous flows. The sequence of these phases could be set differently from one cycle to another. Figure 1 shows possible flows for typical 4-leg, as used by the Tammsaare-Sõpruse intersection. Each leg has a letter that indicates to which side the flow is.

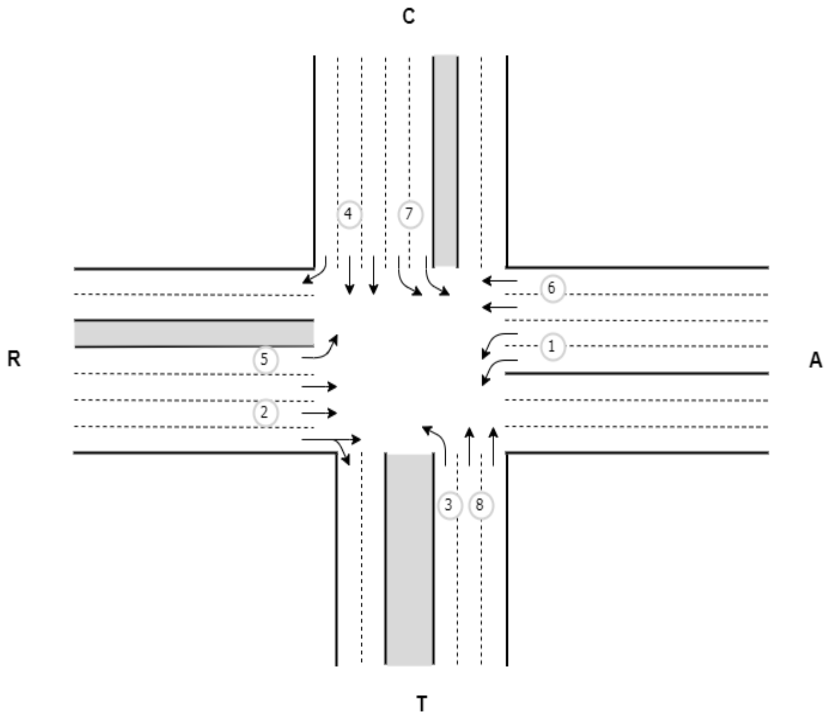


Figure 1. The abstraction of Tammsaare-Sõpruse intersection with possible flows.

It can be seen that vehicles can cross the intersection in three ways: straight, right turn, left turn. The right turn is always coupled with straight, so eight traffic flows could conflict and require a schedule at this intersection. Each flow has a number as an identifier. Table 1 shows possible phases, where flows are described with letters that indicate the origin and destination separated with flow direction (->), e.g., A->T means that vehicle is coming from A and wants to reach T. Origin and destination identifiers are described in Figure 1. As the right turn is coupled with straight, some pair elements have two simultaneous flows.

On the other hand, the non-traditional 4-leg intersection may be more complicated. Each phase may present more than two flows. Figure 2 shows a more complicated intersection. It is an abstraction of the Endla-Paldiski-Mustamäe intersection. In this case, the possible phases are different compared to typical 4-leg intersection, as shown in Table 2. It is complicated to create an algorithm for creating all possible phases as intersections are different.

Furthermore, each phase change needs a practical protection phase. This problem is solved per in-

Table 1. Tammsaare-Sõpruse possible phases.

Pair	Flow 1	Flow 2
P(1,5)	A->T	R->C
P(1,6)	A->T	A->R
P(2,5)	R->A, R->T	R->C
P(2,6)	R->A, R->T	A->R
P(3,7)	T->R	C->A
P(3,8)	T->R	T->C
P(4,7)	C->T, C->R	C->A
P(4,8)	C->T, C->R	T->C

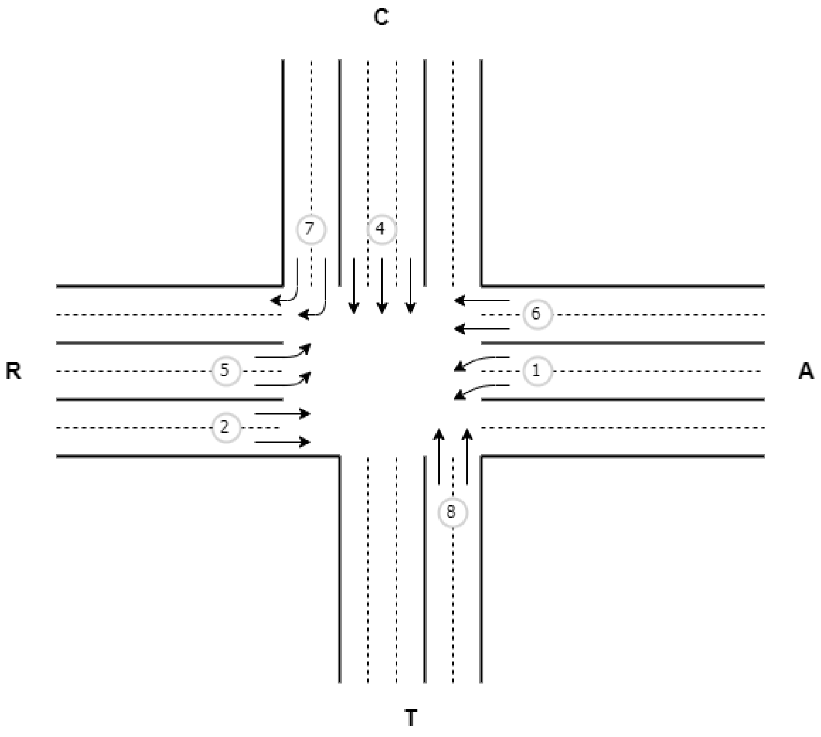


Figure 2. The abstraction of Endla-Paldiski-Mustamäe intersection with possible flows.

tersection. For example, changing phase P1(1,6) to P2(1,5) has non-conflictive flow 1 on a typical 4-leg intersection, nevertheless flow 1 does not need a protection phase, as illustrated in Figure 3. In general, protection phases have to be configured for each transition. Different intersection types make it overly complex to use all of the non-traditional intersection phases optimally in the algorithm. Each intersection is investigated separately. For example, the proposed non-traditional intersection flows can be scheduled once with two tuples and one pair, as a total of 8 flows need to be scheduled. Possible phases for this kind of intersection would be P(2,5,7), P(4,7,8), and P(1,6). In general, it is necessary to schedule all of the flows once during the traffic signal cycle, therefore for traditional 4-leg intersections, four phases out of eight are scheduled each cycle, and for non-traditional intersections, it depends on the configuration of the intersection. Each intersection needs to be independently configured as there is not an optimal abstraction to create traffic phases for all intersections.

Table 2. Endla-Paldiski-Mustamäe possible phases.

Phase	**Phase elements**
P(1,5,7)	A->T, R->C, C->R
P(2,5,7)	R->A, R->C, C->R
P(4,7,8)	C->T, C->R, T->C
P(2,6)	R->A, A->R
P(1,6)	A->T, A->R

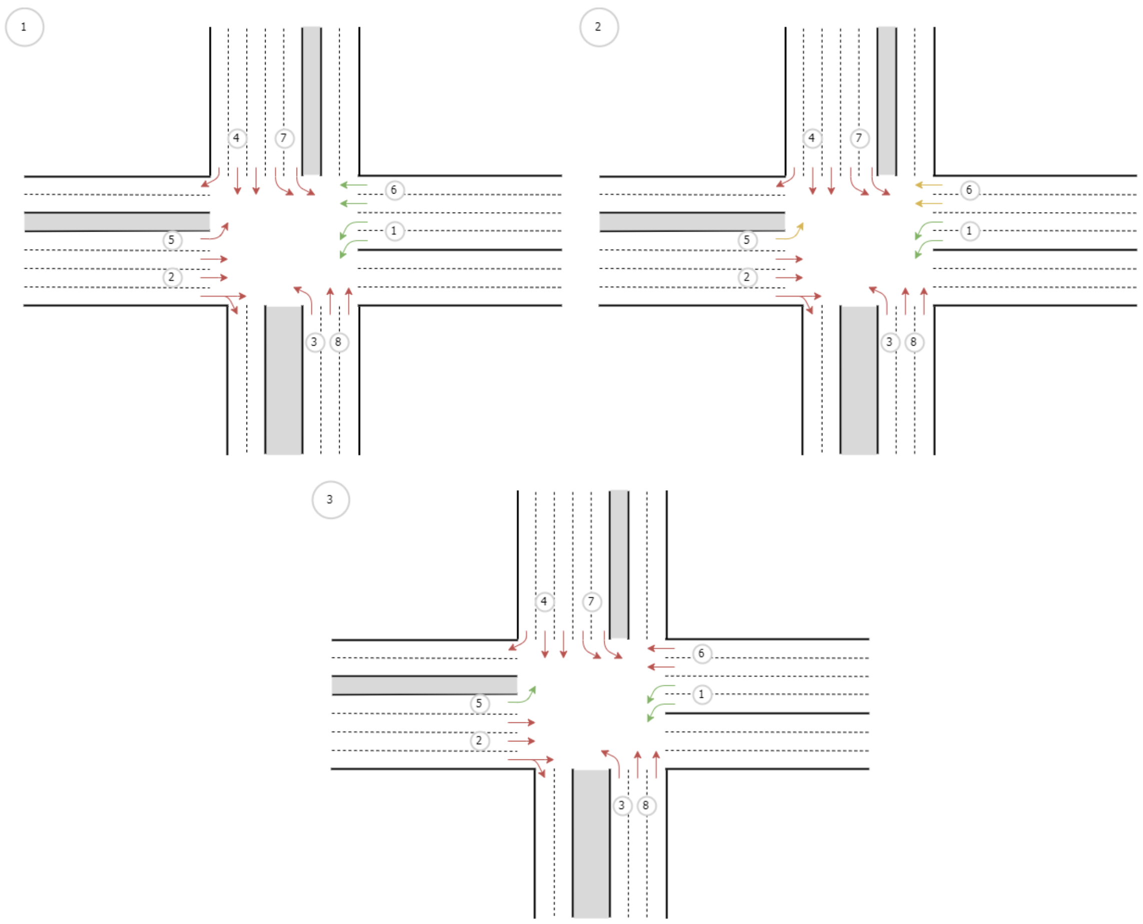


Figure 3. Protection phase going from P(1,6) to P(1,5).

3.2 Priority system

Each phase gets a specific time at each cycle of a traffic signal. The maximum green phase time (*MAX-GREEN*) is set according to the priority level of the flow to maximize the efficiency of the green phase. The priority system is developed offline based on historical public transportation data. In this case, flows with higher public transportation demand would have longer green phases when the traffic is congested. Priority levels should be renewed occasionally when significant changes are made in the public transportation schedule. Table 3, column *Demand per hour*, gives an overview of sample public transportation demand for Tammsaare-Sõpruse intersection using identifiers from Figure 1. The data is from Seire [1] application, showing working days average public transportation demand in February 2020. Flows with zero demand are not included. The average demand per flow is 15 (rounded to the nearest integer) using Equation 1. It makes sense to divide priorities into two levels - level 1 and level 2. Level 1 would be flows with a higher priority and level 2 with lower one. The condition to find level 1 is shown in Equation 2. The condition to find level 2 is shown in Equation 3.

$$M = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{1}$$

$$p = 1 \quad \text{if } d \geq M \tag{2}$$

$$p = 2 \quad \text{if } d < M \tag{3}$$

The priority levels are chosen as in Table 3, column *Priority level*, using the mentioned conditions. The calculated level will choose if priority difference (*MG-DIFF*) will be subtracted from the chosen static maximum green time value or added to it. Static maximum green time is a value that is set as input parameter for the algorithm. It is a base value which will be used for calculation, e.g., if the chosen static maximum green time is 25 seconds and the maximum green time difference is 5 seconds then priority level 2 flows will have 20 seconds (Equation 4) and priority level 1 flows 30 seconds maximum green phase length (Equation 5). Example table of priorities with maximum green times are illustrated in Table 3, using maximum green time equal to 25 and maximum green time difference equal to 5 as inputs.

$$mGreen = smGreen - mgDiff, \quad \text{if } d < M \tag{4}$$

$$mGreen = smGreen + mgDiff, \quad \text{if } d \geq M \tag{5}$$

Table 3. Tammsaare-Sõpruse priority table.

Flow	**Demand per hour**	**Priority level**	***MAX-GREEN* (sec)**
R->A	7	2	20
A->R	7	2	20
A->T	8	2	20
T->C	34	1	30
T->A	5	2	20
T->R	3	2	20
C->T	38	1	30

Furthermore, if the priority system table is calculated, it is assigned inside the algorithm and will stay static. If there is a significant change in the public transportation schedule, the priority table should be recalculated. The actual traffic signal green phase is calculated dynamically. None of

[1] `https://seire.tallinn.ee/`

the flows can have the best green time more than the maximum green time allows.

3.3 Best green time configuration

Dynamic scheduling can calculate green phase length considering the current real-time traffic situation. ITLC [26] and ETLSA [5] have provided an efficient algorithm to find the best green phase length. However, in PETSSA, it is made suitable for Tallinn.

Best green time (*BEST-GREEN*) is selected based on the traffic distribution over the traffic flows of the selected phase. If the maximum green time would always be scheduled, then most vehicles on off-peak hours would pass early, and the intersection would remain empty. On the other hand, the maximum green time is entirely used on-peak hours when the traffic is denser. However, if the best green time is shorter than assigned by the traffic rules of Tallinn, then minimal length (*MIN-GREEN*) is used (Equation 6). In this thesis, the minimal green value is set to 5 which is a common value used in Tallinn. If traffic flow density is zero then green phase is not scheduled, thus the best green time equals to zero (Equation 7).

$$minGreen \leq bestGreen \leq maxGreen, \quad \text{if } density > 0 \tag{6}$$

$$bestGreen = 0, \quad \text{if } density = 0 \tag{7}$$

$$bestGreen = \frac{lvDistance}{tSpeed}, \quad \text{if } density > 0 \tag{8}$$

$$dGreen = maxGreen * tSpeed \tag{9}$$

The best green time is calculated separately for each flow to reduce the pointless time of the green phase. There are three main use-cases:

1. When there are no vehicles near the intersection, thus, the best green time would be zero (Figure 4).
2. When there are vehicles inside the area of maximum green time (Figure 5), then the best green time would be calculated dynamically. The number of vehicles that can pass through the maximum green time is detected based on the traffic distribution of the selected phase, using average traffic speed from the speed sensor. The best green time is then defined as the time required to allow the last vehicle to pass the intersection. Equation 8 computes the *BEST-GREEN* value, where *dGreen* is the *MAX-GREEN* area (Equation 9), and *lvDistance* is the distance between the intersection and the last vehicle of the traffic flow inside *dGreen*, and *tSpeed* is the average traffic speed from the speed sensor.
3. When the flow is congested, the best green time is equal to the max green time, as vehicles

are demanding (Figure 6).

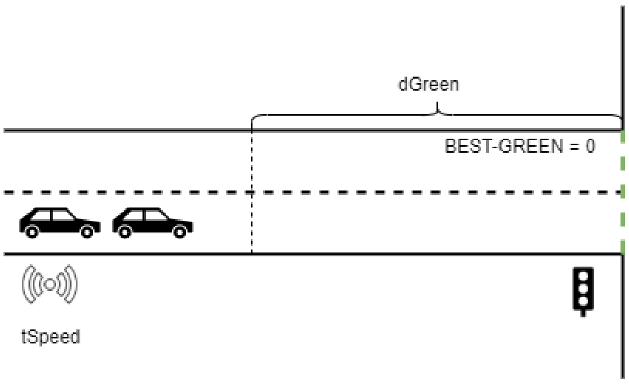


Figure 4. *BEST-GREEN* when no vehicles near intersection.

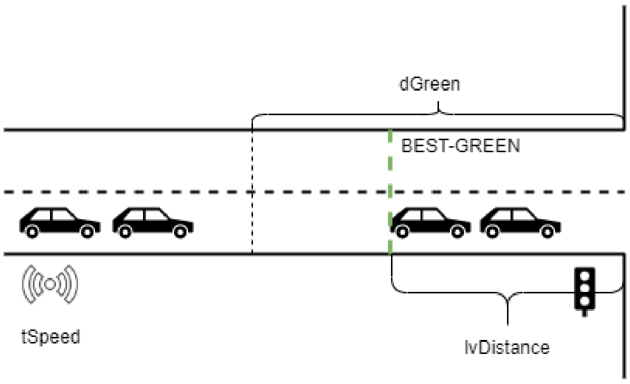


Figure 5. *BEST-GREEN* when vehicles inside *dGreen*.

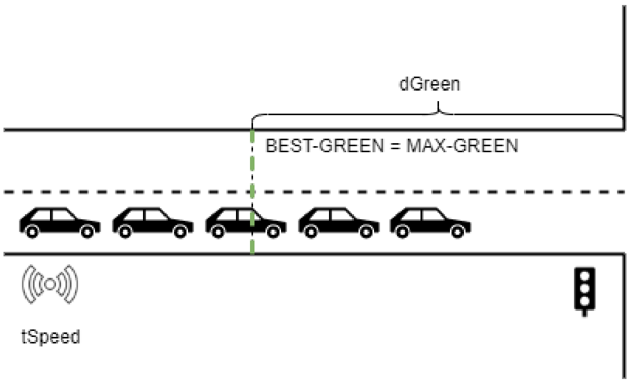


Figure 6. *BEST-GREEN* when flow is congested.

The best green time is calculated for each phase, and when scheduling the phase then the maximum best green time of the phase flows is chosen. Therefore, the longest best green phase length is chosen for the selected traffic signal phase. In general, choosing the best green phase time for each traffic phase makes the algorithm dynamic and efficient to adapt to the current traffic situation. However, the vehicles must be detected precisely around the intersection, which can be done using sensing technologies.

3.4 Camera data and speed sensors

PETSSA needs data about vehicle positions near the intersection and the average speed traffic of the flow. Recent developments in machine learning have made it possible to detect vehicles near intersections effectively using camera video. A study about traffic congestion detection found out that CCTV can be an essential data source for determining the state of traffic congestion [6]. It is also found out that even at poor camera conditions, the accuracy of the model is more than 0.9 for all weather conditions [6]. An example of camera and speed sensor positions is provided in Figure 7. The camera should be able to detect each line data separately. Furthermore, Tallinn University of Technology has installed speed sensors to various places in Tallinn [15]. It is possible to use speed sensors further away from the intersections to predict the average traffic speed.

Detecting vehicles, measuring traffic speed, and parsing the data of camera flow is out of scope in this thesis. In this thesis, we assume that cameras and sensor network is working in an ideal environment, and the data is already parsed to the standard form.

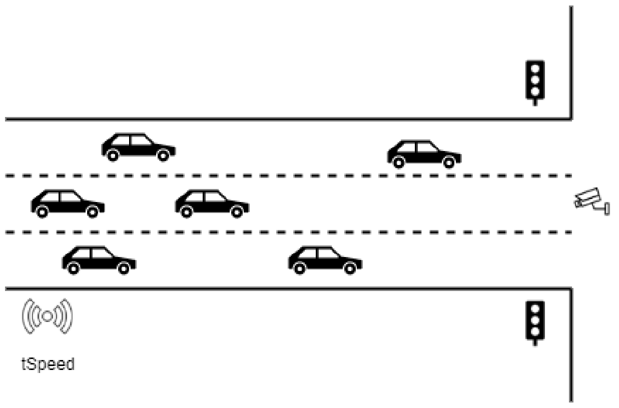


Figure 7. Possible sensors positions.

3.5 The prioritized traffic signal scheduling algorithm

Taking all of the calculated data as inputs, PETSSA creates an efficient and sustainable traffic signal schedule. For a better overview, the pseudo-code of PETSSA's one traffic cycle schedule is illustrated systematically in Algorithm 1. The first step is to find all possible flows (Algorithm 1, line 1), create phases considering the type of intersection and possible phases (Algorithm 1, line 2), and then create effective protection phases between all phases (Algorithm 1, line 3). Next, a configuration input is loaded which contains *static MAX-GREEN*, *MG-DIFF*, *MIN-GREEN*, and priority levels table for each phase. These are parameters that do not change during scheduling traffic signals. Then, based on the priority levels, the maximum green length is calculated (Algorithm 1, line 4). After that, if there are unscheduled flows for the traffic signal cycle, camera data and speed sensors data is parsed to a standard form so that the PETSSA can take it as inputs (Algorithm 1, line 6).

```
Algorithm 1: PETSSA SCHEDULING ONE CYCLE
Input: Intersection configuration $C$, static MAX-GREEN $G_{max}$, MIN-GREEN $G_{min}$,
       MG-DIFF $G_{diff}$, Priority levels $P_{levels}$

1  $F_c \leftarrow$ GET_FLOWS ($C$);
2  $P_f \leftarrow$ GET_POSSIBLE_PHASES ($F_c$, $C$);
3  $P_p \leftarrow$ GET_PROTECTION_PHASES ($P_f$);
4  $M_p \leftarrow$ GET_PHASES_MAXGREEN_VALUES ($P_f$, $G_{max}$, $G_{diff}$, $P_{levels}$);
5  while ( HAS_UNSCHEDULED_FLOWS ($F_c$)=True ) do
6      $S \leftarrow$ GET_SENSOR_DATA();
7      $F_p \leftarrow$ COMPUTE_FLOW_PARAMETERS ($F_c$, $S$);
8      $P_{bg} \leftarrow$ GET_PHASES_BESTGREEN_VALUES ($P_f$, $F_p$, $G_{min}$, $M_p$);
9      if ( EXISTS_NON_ZERO_BESTGREEN ($P_bg$) = True ) then
10         $F_1 \leftarrow$ GET_DEMANDING_UNSCHE_FLOW ($F_p$);
11         $F_2 \leftarrow$ GET_PAIRABLE_DEMANDING_UNSCHE_FLOW_INSIDE_DGREEN ($P_f$,
            $F_p$, $F_1$);
12         if ( IS_ZERO_DENSITY ($F_2$) = True ) then
13             $F_2 \leftarrow$ GET_PAIRABLE_OVERALL_DEMANDING_UNSCHE_FLOW ($P_f$,
                $F_p$, $F_1$);
14         MARK_AS_SCHEDULED ($F_c[F_1]$, $F_c[F_2]$);
15         SCHEDULE_PHASE ($P_f[F_1, F_2]$, $P_p[F_1, F_2]$, $P_{bg}[(P_f[F_1, F_2])]$);
```

Furthermore, based on the traffic speed of each flow, the farthest distance that moving vehicles can traverse past the intersection during this maximum green time is determined (Algorithm 1, line 7). Then, the best green time for each flow is calculated (Algorithm 1, line 8). When the best green time equals zero, there is no density near the intersection, and the phase should be skipped. Each scheduled phase should permit at least two synchronous traffic flows to cross the intersection during the scheduled period of maximum green time. The sequence of phases is selected based on the density of each traffic flow. For each phase, from unscheduled flows, the highest density flow *F1* is chosen (Algorithm 1, line 10). Then, from unscheduled flows, the highest density flow inside the *dGreen* area *F2* is chosen (Algorithm 1, line 11). *F2* must be pairable with *F1*. After that, if F2 density is equal to zero, then from unscheduled flows, the highest overall density flow is chosen (Algorithm 1, line 13). Once again, it must be pairable with *F1*. *P(F1,F2)* should be scheduled for the best green time. *F1* and *F2* are set as scheduled, and scheduling traffic signal to the chosen phase starts with the optimal protection phase (Algorithm 1, lines 14 – 15). After the protection phases, the chosen phase is scheduled. If the phase has more than two flows, then the decision to schedule phase is made considering the two most dense flows. These steps are executed repeatedly until all flows are marked as scheduled in that traffic cycle. After that, the next cycle of the traffic light may be configured. All flows are set to unscheduled, and all flow parameters are reset. Figure 8 provides a data flow for better understanding, where the data comes from and how it moves.

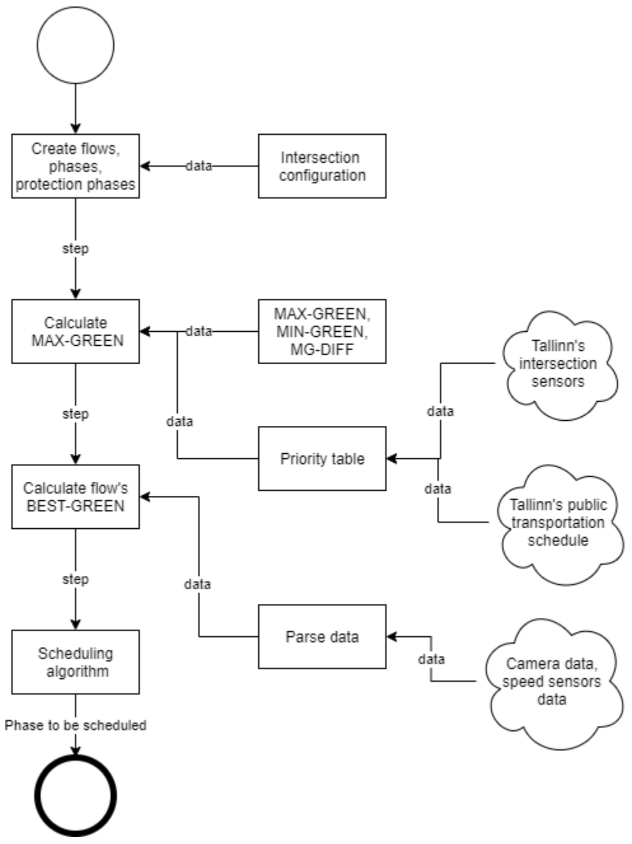

Create flows, phases, protection phases
data
Intersection configuration
step
Calculate MAX-GREEN
data
MAX-GREEN, MIN-GREEN, MG-DIFF
Tallinn's intersection sensors
step
data
data
Priority table
Calculate flow's BEST-GREEN
data
Tallinn's public transportation schedule
step
data
Parse data
data
Scheduling algorithm
Camera data, speed sensors data
Phase to be scheduled

Figure 8. Data flow of PETSSA algorithm.

As for the algorithm result, the cycle length is different each cycle, and if the traffic is congested, then the flows with the highest priority would be cleared quicker, as the maximum green phase length is longer. This approach makes traffic more sustainable and efficient. The performance of proposed algorithm is determined using realistic simulation environment SUMO with Tallinn's intersections.

4 Tallinn’s intersections realistic simulation

In this chapter, the setup of a realistic simulation environment in SUMO is described, as it is crucial to evaluate the performance of the proposed algorithm. The thesis focuses on three key intersections to provide evidence that PETSSA is efficient and sustainable in Tallinn – Tammsaare-Sõpruse, Tammsaare-Mustamäe, and Endla-Paldiski-Mustamäe. It is possible to get realistic simulation results using detailed microscopic simulation, which measures and models each vehicle and its dynamics individually. SUMO is used to simulate traffic networks adding vehicles to travel through intersections. Therefore, SUMO can distinguish different types of vehicles making the simulation precise about cars and public transportation. This chapter presents an overview of three main steps to create comparable realistic simulations – setting up the environment for realistic simulations in SUMO; simulating baseline measures for chosen intersections; integrating and running PETSSA within the simulation environment.

4.1 Environment setup

It is often a time-consuming process to get parameters for realistic traffic simulation. However, SUMO provides a large package of helper scripts [18]. The main parameters to create a realistic simulation are the following:

- Network data (e.g., roads, footpaths);
- Traffic infrastructure (e.g., traffic lights, junction types);
- Traffic demand (e.g., demand for cars, public transportation);
- Simulation parameters (e.g., output logs, simulation length).

Creating a realistic traffic simulation with different traffic signal schedules could be divided into four steps. The process of preparing realistic road networks and infrastructure for Tallinn intersections is described in Sub-section 4.1.1. Gathering traffic demand and creating reproducible simulations is described in Sub-section 4.1.2. Creating and running baseline simulations are described in Section 4.2. The integration and implementation of PETSSA with SUMO are described in Section 4.3.

After gathering the parameters and defining scenarios, it is useful to observe the simulation objects in a visual representation for validation. SUMO provides SUMO-GUI application, which allows observing the simulation at different speeds [18]. Sample SUMO-GUI visualization is shown in Figure 9. To validate the simulation scenario quantitatively, SUMO provides output log files that can be enabled selectively. Some of them are used in this thesis to measure the performance, such as:

- Vehicle trip info (position and speed);
- Traffic data collected from sensors (cameras, speed sensors);
- Vehicle aggregated data (trip duration, emission, fuel consumption);
- Protocols of traffic light switching.

This data can be aggregated using sumo-lib [1] in Python. Version 3.7.7 of Python [2] is used to run this thesis simulation with traffic control in SUMO.

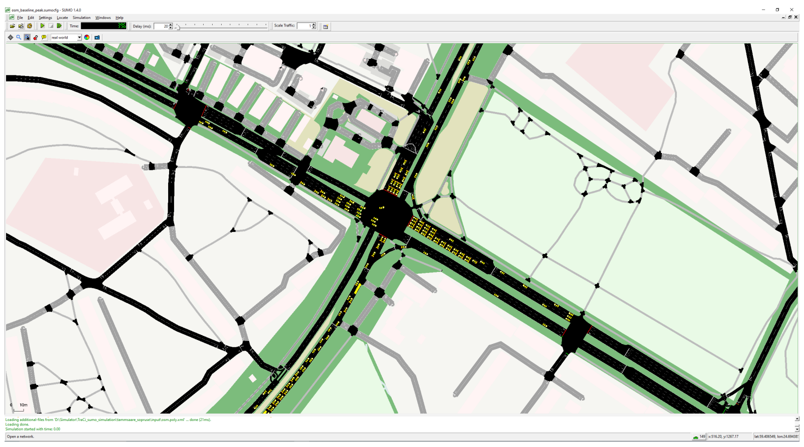


Figure 9. SUMO-GUI showing intersection network.

4.1.1 Network generation

Simulating a realistic scenario requires a realistic traffic network. SUMO network consists of nodes, edges, waterways, tracks, bike lanes, and walkways. Edges represent possible streets. Each edge consists of one or more lanes running in parallel. Many attributes, e.g., speed limit and access permissions, can be added as an edge attribute. SUMO networks are created with NETCONVERT and NETEDIT applications to ensure consistent network format [18]. NETCONVERT

[1] `https://sumo.dlr.de/docs/Tools/Sumolib.html`
[2] `https://www.python.org/`

is a command-line tool that can be used to import networks from different sources, e.g., OpenStreetMap (OSM) [18]. The key feature of NETCONVERT is the heuristic refinement of missing network data to achieve the necessary level of detail for microscopic simulation [18]. Therefore, network is complete, and vehicles can be simulated after generating a new network. A useful tool for initial scenario preparation is SUMO's osmWebWizard. All of the chosen intersections in Tallinn were first exported with osmWebWizard. It uses NETCONVERT and OSM data to generate a network that is needed for the scenario. Browser-based WebWizard is executed, and the needed area of the map is selected and data downloaded from OSM with parameters corresponding to the selected traffic modes. Figure 10 shows sample WebWizard export from Tammsaare-Sõpruse intersection. Random traffic is then generated to populate the network. However, in this thesis, the traffic must be realistic and random traffic is deleted.

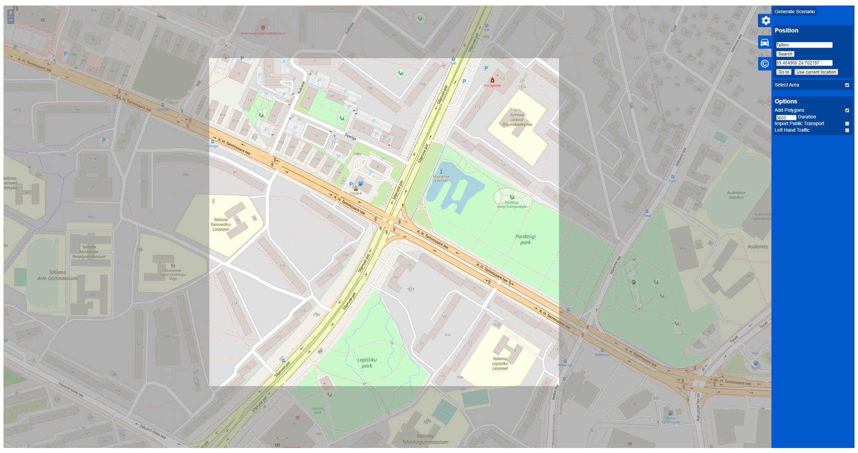


Figure 10. Tammsaare-Sõpruse in SUMO's WebWizard

Due to frequent mismatch between available input data and the necessary level of detail for microscopic simulation, network and infrastructure preparation is often a challenging task. Sometimes OSM data is not well defined and needs additional editing. NETEDIT is a graphical network editor that can be used to create, analyze, and edit network files that supports defining additional traffic infrastructure which could not be imported by NETCONVERT [18]. The supported features for editing network files are as follows [18]:

- Network elements (junctions, edges and lanes);
- Advanced network elements (e.g. traffic signals);
- Additional infrastructure (e.g. bus stops, pedestrian paths);
- Polygons and points of interest (POI).

Furthermore, OSM data about Tallinn's intersections is incomplete. Each OSM exported intersection needs editing, e.g., change the number of lines, recompute junctions, change lines priority, change traffic signal logic. OSM provides a possibility to request a change to a map, but in this thesis, many of the changes were too large, and a simpler way was to edit the exported network with NETEDIT and get correct traffic rules from real-life examination or Google Street View [3]. Figure 11 shows NETEDIT environment editing Tammsaare-Sõpruse intersection. It is crucial to

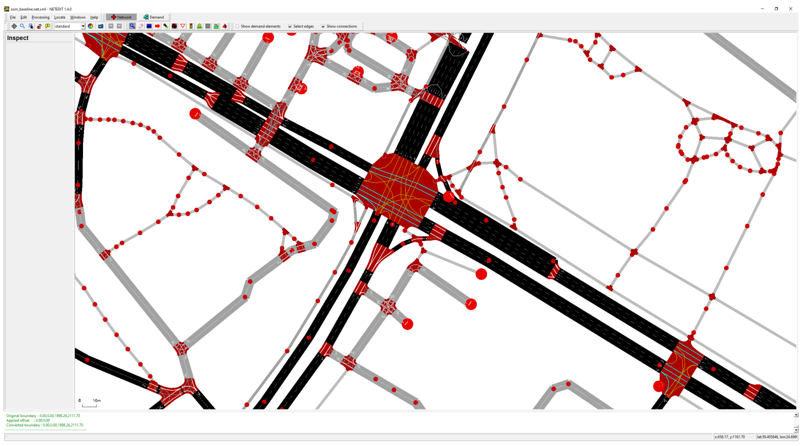


Figure 11. Tammsaare-Sõpruse in NETEDIT.

do visual testing that the edges used in the simulation are correctly edited and represent the real-life situation. The easiest way is to examine the intersection in the Google Street View and make sure the simulation environment is correctly implemented. As the simulation needs to be realistic, random traffic demand is deleted, and realistic traffic demand must be created to make the simulation as realistic as possible. In SUMO environment, vehicles and routes generation means creating a demand XML file.

4.1.2 Vehicles and routes generation

One of the critical parameters to make the simulation more realistic is to provide traffic demand that is extracted from real-life measurements. In SUMO, traffic demand can be defined as individual trips, flows, or as routes. The necessary information should include departure time, origin, destination, and transport mode, e.g., car, public transportation, or pedestrian. SUMO has four tools to generate demand from different data sources - ACTIVITYGEN, Flowrouter, DFROUTER, JTRROUTER, and randomTrips.py script [18]. In this thesis, the demand generation script is made from scratch, as it is necessary that all vehicles travel from origin to destination near the intersection, and the number of vehicles must be the same as in real life peak and off-peak hours. The input data for the number of vehicles come from three sources:

- Seire application;
- Tallinn's public transportation schedule [4];

[3]`https://www.google.com/streetview/`
[4]`https://transport.tallinn.ee/`

- Tallinn University of Technology Logistics and Transport Research Group data about traffic demand [5].

First, the length of the simulation is defined. In this thesis, all simulations are 1 hour long, therefore during 3600 simulation seconds, the simulation produces vehicles from origin into the network. Origin and destination are edge identifiers from the generated network. Traffic demand, origin, and destination edges are different for each simulation, therefore, each intersection needs a different demand preparation script. Origins have to be chosen far enough considering maximum green distance (Section 3.3). Tammsaare-Sõpruse intersection's origins and destinations are illustrated in Figure 12, Tammsaare-Mustamäe's in Figure 13 and Endla-Paldiski-Mustamäe's in Figure 14. Other signalized intersections near the main intersection are disabled, and cars can go through them freely. In this case, the main focus stays on one signalized intersection.

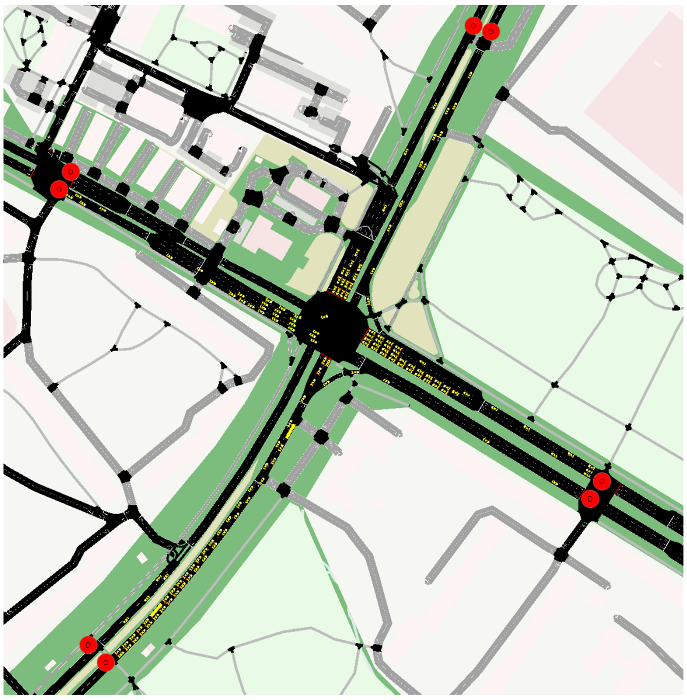


Figure 12. Tammsaare-Sõpruse routes origins and destinations.

Next, the demand for vehicles per hour is determined. Vehicles are divided into two groups - cars and public transportation. For each group, demand and routes are determined separately for peak and off-peak hours. This distinction allows the microscopic simulation to be precise about vehicle types and traffic conditions. Peak hour is considered as 8 am to 9 am on working days and an off-peak hour from 8 pm to 9 pm on working days. Table 4 illustrates demands for each intersection on peak hour and Table 5 on off-peak hour. It is worth mentioning that the demand is much higher on peak hour, and congested situations occur. Furthermore, total demand is divided

[5] https://www.ttu.ee/instituut/mehaanika-ja-toostustehnika-instituut/keskused/logistika-ja-transpordi-teaduskeskus/

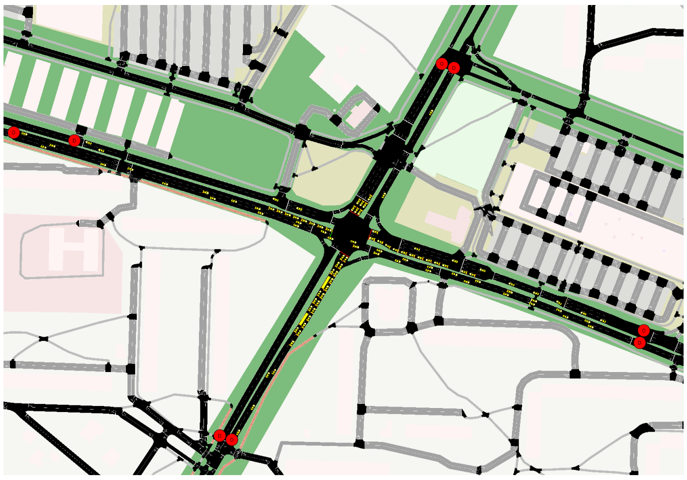


Figure 13. Tammsaare-Mustamäe routes origins and destinations.

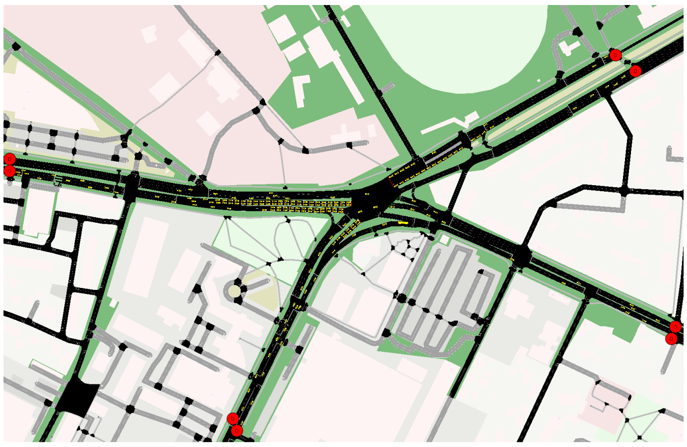


Figure 14. Endla-Paldiski-Mustamäe routes origins and destinations.

Table 4. Traffic demand on peak hour from 8 am to 9 am for chosen intersections

Intersection	**Car demand peak**	**Public transportation demand peak**
Tammsaare-Sõpruse	5485	102
Tammsaare-Mustamäe	4089	81
Endla-Paldiski-Mustamäe	6639	102

Table 5. Traffic demand on off-peak hour from 8 pm to 9 pm for chosen intersections

Intersection	**Car demand off-peak**	**Public transportation demand off-peak**
Tammsaare-Sõpruse	2103	47
Tammsaare-Mustamäe	1638	48
Endla-Paldiski-Mustamäe	2528	61

into origin-destination (OD) groups. Tallinn's sensors provide this kind of exactness at significant intersections. This data is available in the Seire application. An OD matrix is calculated for each intersection. OD matrix provides information about origin and destination edges identifiers and the demand for cars and public transportation to travel between these edges. Trip records are generated randomly, considering the overall needed demand for each OD pair. It ensures that the demand is distributed between the total simulation period. In that case, depart time can vary from 0 to 3600. As cars and public transportation demands are used separately, different demand XML files are made for each transportation group. Vehicle demands are described as *<trip/>* elements that define vehicle id, type, depart time, depart lane, origin edge, and destination edge. Sample demand XML file is illustrated in Code example 4.1, which shows a sample car trip from origin to destination and Code example 4.2 illustrates sample public transportation trip.

Once traffic demand is generated, traffic assignment algorithm can be executed to understand the traffic state of the network. In this thesis, the fastest route at a given departure time algorithm is used. The route is calculated in SUMO directly, using the Dijkstra algorithm, which is the simplest and slowest of routing algorithms. This routing takes into account the current and recent state of traffic in the network and thus adapts to jams and other changes. For automatic routing, default settings are used. After providing vehicle demand XML files to SUMO network configuration, the simulation is ready to run with the default traffic signal logic.

```xml
<routes>
  <vType id="veh_passenger" vClass="passenger"/>
  <trip id="car_tty_centre_1763" type="veh_passenger" depart="974"
    departLane="best" from="7842435#3" to="78266149#5"/>
</routes>
```

Code example 4.1. Example of cars demand XML file.

```xml
<routes>
  <vType id="bus_bus" vClass="bus"/>
  <trip id="bus_rocca_centre_26" type="bus_bus" depart="800" departLane
    ="best" from="49689069#2" to="78266149#5"/>
</routes>
```

Code example 4.2. Example of public transportation demand XML file.

In addition, to make simulation realistic and PETSSA comparable to the current situation, it is necessary to determine baseline simulation results, which means simulating with realistic demand and traffic signal schedule.

4.2 Running baseline simulation

Baseline simulations are necessary measures to compare introduced PETSSA performance to realistic traffic scenarios. Both simulations are run in the same SUMO environment. However, the baseline traffic signal schedule is using realistic phases that have been measured during March

2020 at midday. For a better understanding of the baseline signal schedule, each signalized line has been assigned a number (Figure 15). In SUMO, each lane traffic signal is scheduled inde-

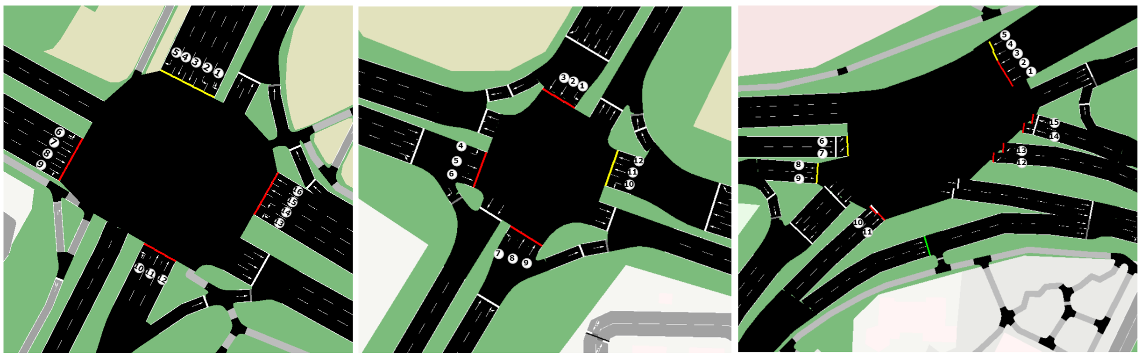


Figure 15. (a) Tammsaare-Sõpruse, (b) Tammsaare-Mustamäe and (c) Endla-Paldiski-Mustamäe lane numbers for traffic signal scheduling logic.

pendently, and each phase has a certain length. Each lane signal is defined as red (R), yellow (Y), or green (G) to make phases understandable. Real-life traffic signal logic for Endla-Paldiski-Mustamäe intersection is shown in Table 6 and Tammsaare-Sõpruse, Tammsaare-Mustamäe intersections in Table 7.

Table 6. Endla-Paldiski-Mustamäe baseline traffic signal logic.

Phase length (s)	**Phase (lines 1-15)**
2	RRRYYYYYYRRRRRR
18	RRRGGGGGGRRRRRR
2	RRRGGYYYYRRRRRR
2	YYYGGRRRRYYRRRR
6	GGGGGRRRRGGRRRR
2	GGGYYRRRRGGRRRR
20	GGGRRRRRRGGRRRR
2	YYYRRRRRRYYRRRR
2	RRRRRRRRRRRYYYY
28	RRRRRRRRRRRGGGG
2	RRRRRRRRRRRYYYY

Mentioned traffic signal phases are converted to SUMO's Traffic Light Schedule (TLS) and added to the baseline network XML file. A total of two baseline simulations are run for each intersection – one with peak hour demand and another with off-peak hour demand. Each simulation produces a trip information log file that provides vehicles delay time, trip duration, fuel consumption. Summary generation script is executed after simulation, which parses each transportation group key measures from log files:

- Delay time;
- Trip duration;

- Fuel consumption;
- Average time for 1 person delivery from origin to destination.

Table 7. (a) Tammsaare-Sõpruse and (b) Tammsaare-Mustamäe intersections baseline traffic signal logic.

Phase length (s)	**Phase (lines 1-16)**	**Phase length (s)**	**Phase (lines 1-12)**
2	YYYYYRRRRRRRRRRR	2	RRRRRRRRRYYY
8	GGGGGRRRRRRRRRRR	8	RRRRRRRRRGGG
2	YYGGGRRRRRRRRRRR	2	RRRRRRRRRYGG
2	RRGGGRRRRRYYRRRR	2	RRRRYYRRRRGG
3	RRGGGRRRRRGGRRRR	18	RRRRGGRRRRGG
2	RRYYYRRRRRGGRRRR	2	RRRRGGRRRRYY
2	RRRRRRRRRYGGRRRR	2	RRRYGGRRRRRR
8	RRRRRRRRRGGGRRRR	8	RRRGGGRRRRRR
2	RRRRRRRRRYYYRRRR	2	RRRYYYRRRRRR
2	RRRRRYYYYRRRRRRR	2	RRRRRRYYYRRR
8	RRRRRGGGGRRRRRRR	8	RRRRRRGGGRRR
2	RRRRRYGGGRRRRRRR	2	RRRRRRYGGRRR
2	RRRRRRGGGRRRRRYY	2	RYYRRRRGGRRR
8	RRRRRRGGGRRRRRGG	3	RGGRRRRGGRRR
2	RRRRRRYYYRRRRRGG	2	RGGRRRRYYRRR
2	RRRRRRRRRRRRYYGG	2	YGGRRRRRRRRR
8	RRRRRRRRRRRRGGGG	13	GGGRRRRRRRRR
2	RRRRRRRRRRRRYYGG	3	YYYRRRRRRRRR
3	RRRRRRRRRRRRRRGG		
2	RRRRRRRRRRRRRRYY		

Results are provided in performance evaluation (Section 5), where baseline results are compared to PETSSA simulation results. PETSSA needs to be integrated into the simulation environment to get the results with the proposed dynamic signal schedule.

4.3 Integrating and running simulation with PETSSA

The evaluation of developed traffic signal programs, to make traffic lights adaptable to the current traffic situation, is one of the main applications for microscopic traffic flow simulations. With Traffic Control Interface (TraCi), the traffic and network simulators are connected in real-time by TraCI, thus enabling the control of mobility attributes of each simulated vehicle near the intersection. TraCi uses Transmission Control Protocol (TCP) based client-server architecture to provide access to SUMO. Thereby, SUMO acts as a server that is started with a command-line option that provides the port in which SUMO would listen for incoming connections. When SUMO is started in remote port mode, it only prepares the simulation network and waits for external applications to connect to the server and take over the control. The client has to trigger each simulation step in SUMO and can ask vehicle data around intersections. If any subscriptions have been done,

```
python %SUMO_HOME%\tammsaare_sopruse_runner.py --nogui --max-green 25
    --mg-diff 5 --type peak
```

Code example 4.3. Command to run automated Tammsaare-Sõpruse intersection simulation with PETSSA.

then the subscribed values are returned as well. In this thesis, the client is a Python program, which controls the simulation and asks information from SUMO, while providing the PETSSA algorithm with the needed information and letting PETSSA make the next traffic signal choice. Simulation client uses the *traci* Python package as an abstraction to the TCP calls that the TraCi makes through the command line. PETSSA is also implemented in Python and is decoupled from the simulation. It only needs input parameters to return the next phase of the traffic signal.

Furthermore, the simulation is started from the command line by the user. Each simulation client describes all possible flows with priorities, and phases containing synchronous flows. Running simulation client needs *MAX-GREEN*, *MG-DIFF*, and simulation type as input. *MIN-GREEN* is universally set to 5 seconds. Sample command to run Tammsaare-Sõpruse intersection simulation is provided in Code example 4.3.

Each simulation runner then executes SUMO server and waits for a response. Once the server is running, TraCi can take over and give commands to the server. Each simulation step is then triggered and checked if the traffic signal needs a new phase as the current phase could still be active. If a new phase must be chosen, the simulation client requests information from the SUMO server, parses it to a common form, and sends it to PETSSA with a request to choose a new phase. PETSSA returns the new chosen phase, and the client once again sends information to the server. Once no vehicles are on the simulation network, the simulation can be stopped. If no vehicles are in the neighborhood of the intersection, then all red signals should be scheduled. After a successful run, a summary is generated based on output log files using the same measures as baseline simulation and returned to the user. Abstraction of simulation sequence is illustrated in Figure 16. This simulation sequence is executed for each intersection to get simulation results. The results are compared to baseline results to validate the efficiency of PETSSA.

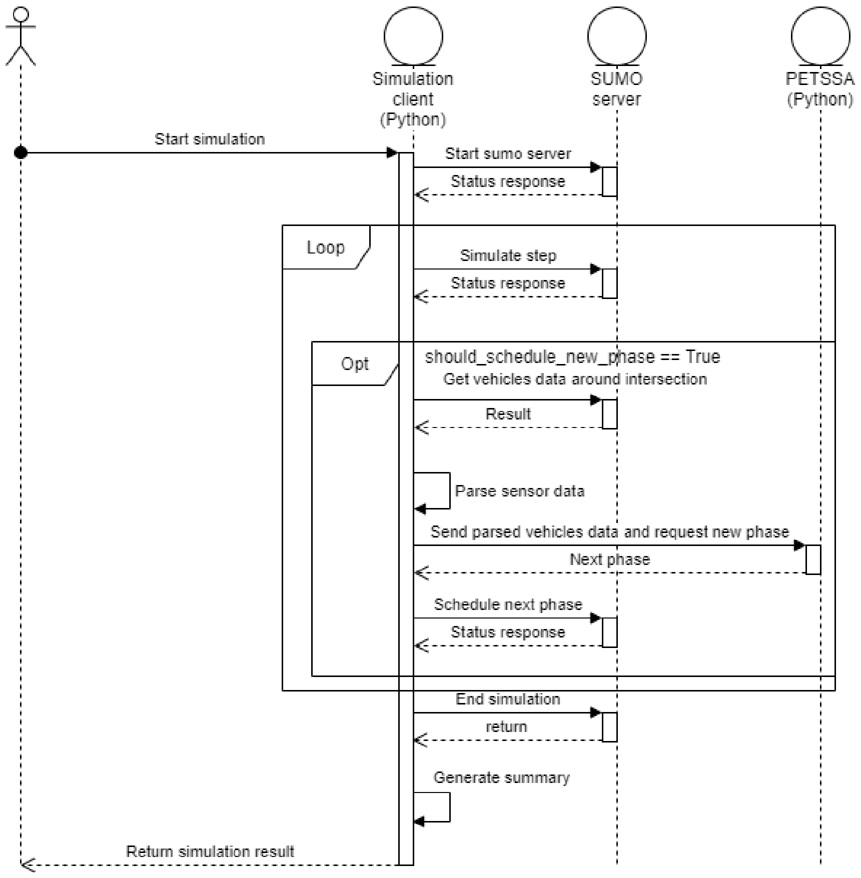


Figure 16. Simulation sequence with Simulation client, SUMO and PETSSA.

5 Perfromance evaluation

In this section, the performance of PETSSA is compared to baseline measures in three intersections in Tallinn – Tammsaare-Sõpruse, Tammsaare-Mustamäe, and Endla-Paldiski-Mustamäe. Both baseline and PETSSA simulations run on the same network environment in SUMO with a different traffic signal schedule. Baseline uses a traffic signal schedule that was measured in real-life, March 2020. Several simulations are run on each intersection to find the best performing input parameters for PETSSA, which ensures that the best traffic signal configuration is chosen for each traffic scenario and intersection. Table 8 illustrates configuration parameters that are executed for all three intersections for both peak and off-peak scenarios. *MIN-GREEN* is set to 5 in all simulations.

Table 8. Configuration input parameters for PETSSA.

***MAX-GREEN* (s)**	***MG-DIFF* (s)**
10	2, 4, 6, 8
15	2, 4, 6, 8
20	2, 4, 6, 8
25	2, 4, 6, 8
30	2, 4, 6, 8
35	2, 4, 6, 8

Key measures to evaluate the performance are average vehicle delay time, average trip duration, and average fuel consumption. It is crucial to mention that the simulations do not consider pedestrians and support the perfect environment in which weather conditions are excellent, and sensors provide no false information. In addition, no traffic incidents involved. Phase configurations and results of each intersection are provided in the following sections.

5.1 Tammsaare-Sõpruse intersection

Tammsaare-Sõpruse intersection is similar to typical 4 leg intersection (Figure 17). Thus, a total of 8 phases can be constructed, each representing a flow pair of synchronous flows. All of the possible phases are following: P(1;5), P(1;6), P(2;5), P(2;6), P(3;7), P(3;8), P(4;7), P(4;8). Only 4 of these phases must be scheduled each traffic signal cycle to allow each flow to be scheduled once.

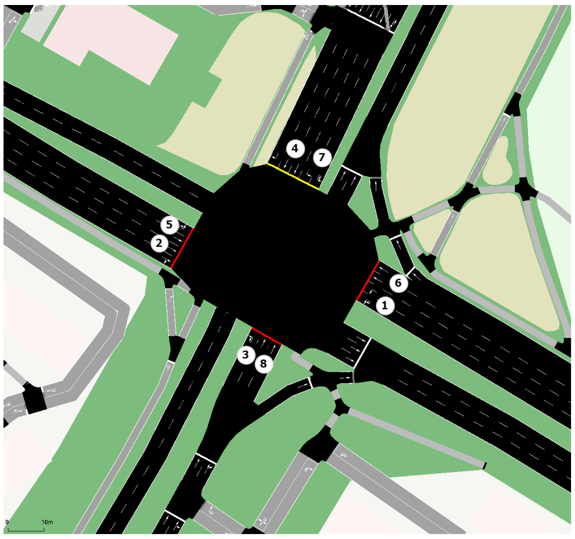
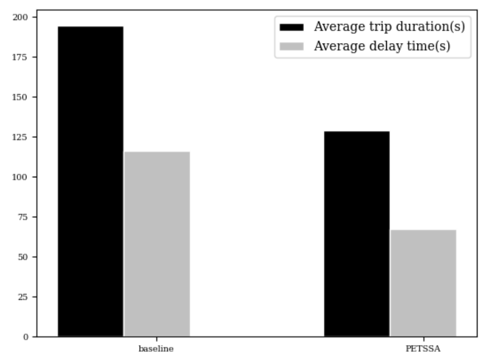


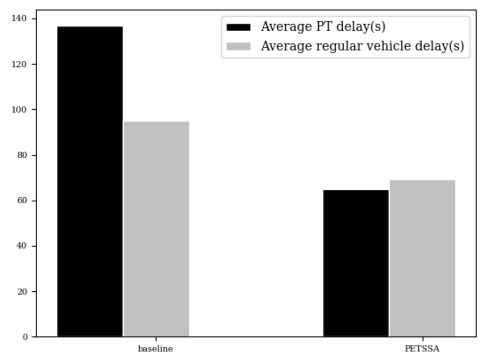
Figure 17. Tammsaare-Sõpruse intersection with possible flows.

On peak hour, PETSSA is very powerful. However, regular vehicles and public transportation performance tend to respond differently to PETSSA configurations. For regular vehicles, the best performing configuration uses *MAX-GREEN* = 30 and *MG-DIFF* = 6 parameters. It can be concluded that regular vehicles respond better with longer green phase length in peak hours. However, public transportation performance tends to respond better to shorter green phases. The best performing configuration uses *MAX-GREEN* = 10 and *MG-DIFF* = 8. Overall performance measures of vehicles is used to find a compromise between best algorithm parameters. The best overall configuration is *MAX-GREEN* = 30 and *MG-DIFF* = 4, which reduces average trip duration by 65 seconds. Each vehicle, during the peak hour, would reach their destination quicker while spending 49 seconds less on the delay time. During the peak hour, a total of 168.0 liters of fuel is consumed less. Figure 18a illustrates average trip duration and average delay time differences compared to baseline. In addition to overall performance improvement, PETSSA prioritizes public transportation over regular vehicles. Figure 18b illustrates how powerful PETSSA is prioritizing public transportation while also reducing regular vehicles delay time. Public transportation delay is reduced by 52.42%, while regular vehicles delay is reduced by 27.35%.

Furthermore, the off-peak hour situation in Tammsaare-Sõpruse intersection is rather good. With a realistic demand on an evening off-peak hour, no congested situation occurs, and optimizing is difficult. PETSSA still manages to shorten all vehicles' average trip duration by 6 seconds. The best configuration uses *MAX-GREEN* = 10 and *MG-DIFF* = 2, which means that phases are rather quick and the difference between priority levels are small. Furthermore, the public transportation delay time is decreased by 43% while regular vehicles delay time is decreased by 23%. Figure 19 illustrates the changes in delay times for different vehicle groups. It can be seen that overall performance is better than baseline, and PETSSA prefers public transportation over



(a) Trip duration and delay time evaluation.



(b) Trip duration and delay time evaluation by vehicle group.

Figure 18. Tammsaare-Sõpruse peak hour performance evaluation.

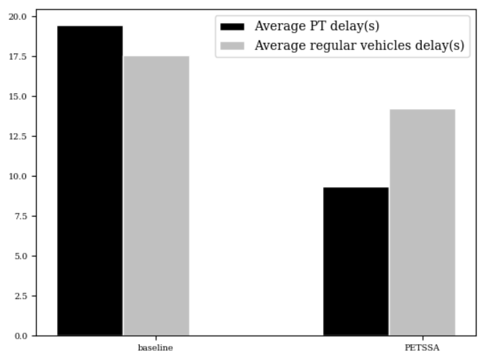


Figure 19. Tammsaare-Sõpruse off-peak hour performance evaluation.

regular vehicles.

In general, based on the simulation results, it is worth mentioning that PETSSA works well with congested and non-congested situations on Tammsaare-Sõpruse intersection. Moreover, public transportation is prioritized while improving overall traffic performance significantly. The overall delay time is reduced, trip time shortened while fuel consumption decreased. Table 9 gives overview of PETSSA performance parameters compared to baseline measures.

Table 9. Tammsaare-Sõpruse PETSSA results compared to baseline on peak and off-peak hours.

Scenario, configuration	**ATD(%)** [1]	**AD(%)** [2]	**AFC(%)** [3]
Peak, MAX-GREEN=30, MG-DIFF=4	33.68	42.13	17.3
Off peak, MAX-GREEN=10, MG-DIFF=2	8.75	31.36	4.86

[1] Average trip duration decrease
[2] Average delay decrease
[3] Average fuel consumption decrease

5.2 Tammsaare-Mustamäe intersection

Tammsaare-Mustamäe is a 4 leg intersection (Figure 20) with all of right turns open and not scheduled by a traffic signal. A total of 8 phases can be constructed, each representing a flow pair of synchronous flows. All of the possible phases are following: P(1;5), P(1;6), P(2;5), P(2;6), P(3;7), P(3;8), P(4;7), P(4;8). As a regular 4 leg intersection, only 4 of these phases must be scheduled each traffic signal cycle to allow each flow to be scheduled once.

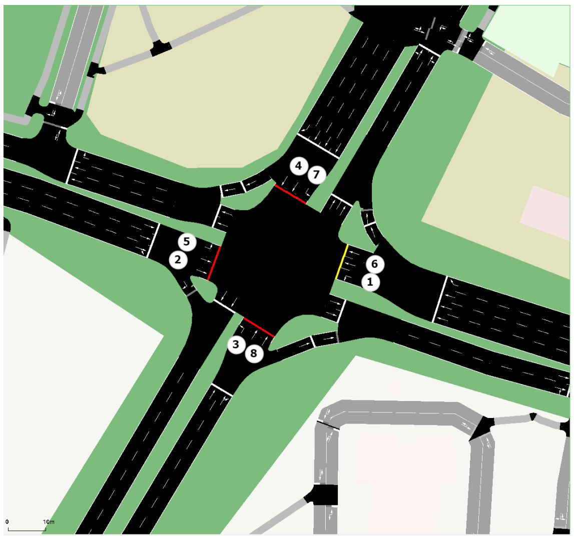
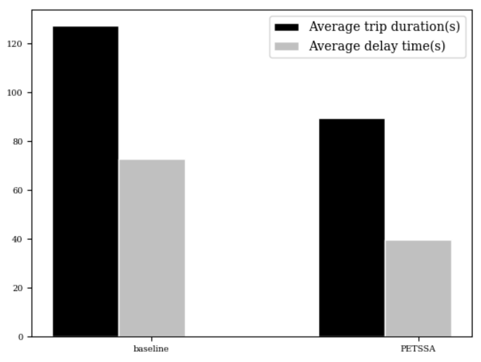


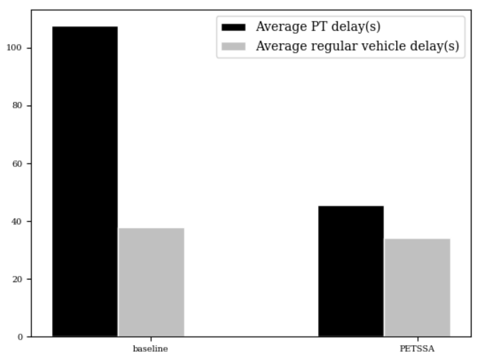
Figure 20. Tammsaare-Mustamäe intersection with possible flows.

On peak hour, PETSSA gives excellent results. Regular vehicles and public transportation have different configurations for the best results. Regular vehicles tend to perform better with shorter maximum green length, as the best configuration uses *MAX-GREEN* = 20 and *MG-DIFF* = 6. However, public transportation performs the best with *MAX-GREEN* = 35 and *MG-DIFF* = 6. Making a compromise and considering all vehicles, PETSSA with *MAX-GREEN* = 25 and *MG-DIFF* = 4 gives the best results overall. It shortens the average trip duration by 37 seconds. Each vehicle would reach their destination quicker while spending 33 seconds less on the delay time. During peak hours, a total of 22 liters of fuel are consumed less. Figure 21a illustrates overall peak hour average trip duration and delay time differences. It can be seen that while prioritizing public transportation, the overall traffic is improved significantly as well. Furthermore, with this configuration, public transportation average delay time decreases 58% while car delay time decreases by 10%. Figure 21b illustrates how powerful PETSSA is for public transportation while still reducing regular vehicles' delay time.

On the off-peak hour, PETSSA also shows excellent results. With realistic demand on an evening off-peak hour, no congested situation occurs. Both public transportation and regular vehicles tend to perform better with shorter maximum green signal phase, as the overall best configuration is *MAX-GREEN* = 10 and *MG-DIFF* = 6. PETSSA manages to reduce the average trip duration by



(a) Trip duration and delay time evaluation.



(b) Trip duration and delay time evaluation by vehicle group.

Figure 21. Tammsaare-Mustamäe peak hour performance evaluation.

15%, while public transportation delay time is reduced by 64%. Figure 22 illustrates the power of prioritized dynamic scheduling that decreases public transportation delay time drastically.

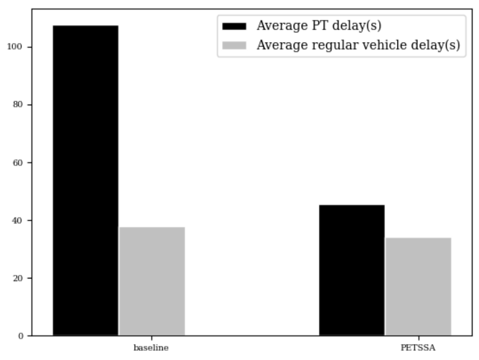


Figure 22. Tammsaare-Mustamäe off-peak hour performance evaluation.

In general, based on the simulation results, it is worth mentioning that PETSSA works well with congested and non-congested situations on Tammsaare-Mustamäe intersection. Public transportation is prioritized while improving overall traffic performance significantly. Even though regular vehicles delay time is not reduced very much, the overall delay time is reduced, trip time shortened while fuel consumption decreased. Table 10 gives overview of PETSSA performance parameters compared to baseline.

Table 10. Tammsaare-Mustamäe PETSSA results compared to baseline on peak and off-peak hours.

Scenario, configuration	**ATD(%)**	**AD(%)**	**AFC(%)**
Peak, MAX-GREEN=25, MG-DIFF=4	29.72	45.20	5.40
Off peak, MAX-GREEN=10, MG-DIFF=6	14.58	41.18	3.33

5.3 Endla-Paldiski-Mustamäe intersection

Endla-Paldiski-Mustamäe intersection (Figure 23) is not a regular 4-leg intersection compared to Tammsaare-Sõpruse and Tammsaare-Mustamäe. Each phase may present more than two flows. All of the possible flows are the following: P(1;5;7), P(2;5;7), P(4;7;8), P(2;6), P(1;6). The goal is to schedule all flows once for each traffic cycle. Therefore following three phases are chosen: P(2;5;7), P(4;7;8), and P(1;6). Each traffic signal cycle would schedule these three phases once.

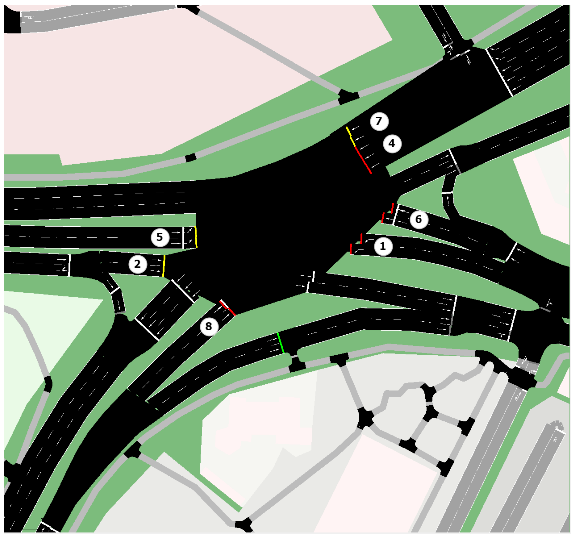
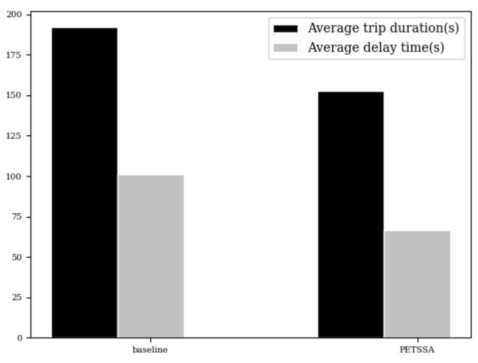
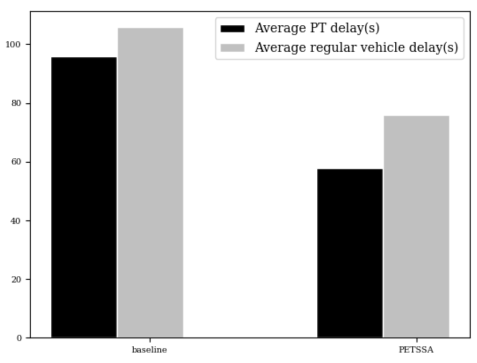


Figure 23. Endla-Paldiski-Mustamäe intersection with possible flows.

On peak hour, PETSSA gives excellent results for both regular cars and public transportation. Both vehicle types tend to have better results with longer maximum length phases, as the best configuration for regular vehicles uses *MAX-GREEN* = 35 and *MG-DIFF* = 4 while best public transportation configuration uses *MAX-GREEN* = 30 and *MG-DIFF* = 4. The best overall configuration is *MAX-GREEN* = 30 and *MG-DIFF* = 4. It shortens the average trip duration by 39 seconds. Each vehicle, during the peak hour, reaches its destination quicker while spending 34 seconds less on the delay time. A total of 210 liters of fuel is consumed less. Figure 24a illustrates overall peak hour average trip duration and delay time differences. In addition to overall performance improvement, PETSSA prioritizes public transportation over regular vehicles. Figure 24b illustrates how powerful PETSSA is for both vehicle groups considering delay time. Public transportation delay is reduced by 40%, while regular vehicles' delay is reduced by 28%. Furthermore, the off-peak hour in Endla-Paldiski-Mustamäe intersection is rather quiet. No congested situation occurs. PETSSA still manages to shorten all vehicles' average trip duration by 11 seconds. Off-peak traffic signal schedule works the best with shorter maximum green phase lengths and smaller differences between priority levels. The best configuration uses *MAX-GREEN* = 10 and *MG-DIFF* = 2. Furthermore, the public transportation delay time is decreased by 55% while regular vehicles' delay time is decreased by 45%. Figure 25 illustrates the changes in delay times for different



(a) Trip duration and delay time evaluation.



(b) Trip duration and delay time evaluation by vehicle group.

Figure 24. Endla-Paldiski-Mustamäe peak hour performance evaluation.

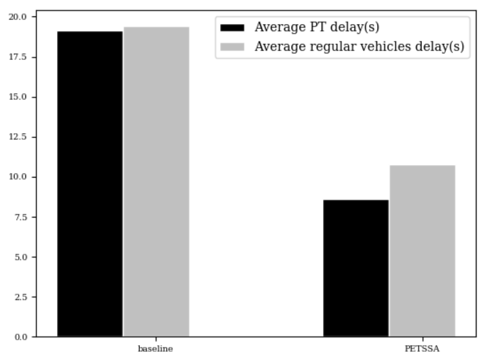


Figure 25. Endla-Paldiski-Mustamäe off-peak hour performance evaluation.

vehicle groups.

In general, it can be said that PETSSA works well with congested and non-congested situations on Endla-Paldiski-Mustamäe intersection. Moreover, public transportation is prioritized while improving overall traffic performance significantly. The overall delay time is reduced, trip time shortened while fuel consumption decreased. Table 11 gives an overview of PETSSA performance parameters compared to baseline measures.

Table 11. Endla-Paldiski-Mustamäe PETSSA results compared to baseline on peak and off-peaks hours.

Scenario, configuration	**ATD(%)**	**AD(%)**	**AFC(%)**
Peak, MAX-GREEN=30, MG-DIFF=4	20.52	33.81	14.73
Off peak, MAX-GREEN=10, MG-DIFF=2	12.18	49.73	10.12

6 Conclusion

In this thesis, a Priority-driven Enhanced Traffic Scheduling Algorithm was introduced, which is designed for Tallinn's intersections to allow more dense traffic flows to cross the intersection first while considering passive public transportation priority levels. The goal was to introduce an algorithm that would improve overall traffic flow while giving more priority to public transportation vehicles. The algorithm works with sensing technologies, such as cameras and speed sensors.

The evaluation of the algorithm needed a realistic simulation environment for Tallinn's intersections. Three major intersections in Tallinn were used to evaluate the performance – Tammsaare-Sõpruse, Tammsaare-Mustamäe, and Endla-Paldiski-Mustamäe. For each intersection, the simulation environment was prepared with realistic traffic network and vehicle demands.

Each simulation network was integrated with baseline traffic signal logic to evaluate the baseline performance. Then, the networks were integrated with introduced PETSSA to evaluate and compare the performance to baseline performance. Simulations were run for both peak and off-peak hours. The simulations showed that PETSSA gives excellent results for both congested and non-congested scenarios. Each evaluated intersection had great results showing that vehicle average trip duration, delay time, and fuel consumption decreased drastically compared to the baseline simulations.

For further development, the algorithm could be improved to be used with pedestrians crossing. In general, traffic is a networking problem, and the evaluation of the performance could be done on a larger scale, using multiple intersections in one simulation.

References

[1] Adu-Gyamfi, Y. O., Asare, S. K., Sharma, A., and Titus, T. (2017). Automated vehicle recognition with deep convolutional neural networks. *Transportation Research Record*, 2645(1):113–122.

[2] Asmaa, O., Mokhtar, K., and Abdelaziz, O. (2013). Road traffic density estimation using microscopic and macroscopic parameters. *Image and Vision Computing*, 31(11):887–894.

[3] Azimirad, E., Pariz, N., and Sistani, M. B. N. (2010). A novel fuzzy model and control of single intersection at urban traffic network. *IEEE Systems Journal*, 4(1):107–111.

[4] Balcilar, M. and Sonmez, A. C. (2008). Extracting vehicle density from background estimation using kalman filter. In *2008 23rd International Symposium on Computer and Information Sciences*, pages 1–5. IEEE.

[5] Bani Younes, M. and Boukerche, A. (2017). An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wireless Networks*, 24.

[6] Chakraborty, P., Adu-Gyamfi, Y. O., Poddar, S., Ahsani, V., Sharma, A., and Sarkar, S. (2018). Traffic congestion detection from camera images using deep convolution neural networks. *Transportation Research Record*, 2672(45):222–231.

[7] Chen, B. and Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):485–497.

[8] Dezani, H., Marranghello, N., and Damiani, F. (2014). Genetic algorithm-based traffic lights timing optimization and routes definition using petri net model of urban traffic flow. *IFAC Proceedings Volumes*, 47(3):11326–11331.

[9] Fellendorf, M. (1994). Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. In *64th Institute of Transportation Engineers Annual Meeting*, volume 32, pages 1–9. Springer.

[10] Ferreira, M. and d'Orey, P. M. (2011). On the impact of virtual traffic lights on carbon emissions mitigation. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):284–295.

[11] Ghaffarian, H., Fathy, M., and Soryani, M. (2012). Vehicular ad hoc networks enabled traffic controller for removing traffic lights in isolated intersections based on integer linear programming. *IET intelligent transport systems*, 6(2):115–123.

[12] Goncalves, W. N., Machado, B. B., and Bruno, O. M. (2012). Spatiotemporal gabor filters: a new method for dynamic texture recognition. *arXiv preprint arXiv:1201.3612*.

[13] Guangwei, Z., Albert, G., and Sherr, L. D. (2007). Optimization of adaptive transit signal priority using parallel genetic algorithm. *Tsinghua Science and Technology*, 12(2):131–140.

[14] Horni, A., Nagel, K., and Axhausen, K. W. (2016). *The multi-agent transport simulation MATSim*. Ubiquity Press London.

[15] Kaugerand, J., Ehala, J., Mõtus, L., and Preden, J.-S. (2018). Time-selective data fusion for in-network processing in ad hoc wireless sensor networks. *International Journal of Distributed Sensor Networks*, 14(11):1550147718811302.

[16] Krauß, S. (1998). *Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics*. PhD thesis.

[17] Li, L., Wen, D., and Yao, D. (2014). A survey of traffic control with vehicular communications. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):425–432.

[18] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y.-P., Hilbrich, R., Lucken, L., Rummel, J., Wagner, P., and WieBner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE.

[19] Ozkurt, C. and Camci, F. (2009). Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks. *Mathematical and Computational Applications*, 14(3):187–196.

[20] P. Dhivya, S. Anitha, K. R. g. R. K. (2016). Smart traffic light junction management using wireless sensor networks. *International Journal of Communication and Networking System*, 5.

[21] Pandit, K., Ghosal, D., Zhang, H. M., and Chuah, C. (2013). Adaptive traffic signal control with vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 62(4):1459–1471.

[22] Rachmadi, M. F., Al Afif, F., Jatmiko, W., Mursanto, P., Manggala, E., Ma'sum, M. A., and Wibowo, A. (2011). Adaptive traffic signal control system using camera sensor and embedded system. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 1261–1265. IEEE.

[23] Skabardonis, A. (2000). Control strategies for transit priority. *Transportation Research Record*, 1727(1):20–26.

[24] Tielert, T., Killat, M., Hartenstein, H., Luz, R., Hausberger, S., and Benz, T. (2010). The impact of traffic-light-to-vehicle communication on fuel consumption and emissions. In *2010 Internet of Things (IOT)*, pages 1–8. IEEE.

[25] Younes, M. B. and Boukerche, A. (2014). An intelligent traffic light scheduling algorithm through vanets. In *39th Annual IEEE Conference on Local Computer Networks Workshops*, pages 637–642.

[26] Younes, M. B. and Boukerche, A. (2016). Intelligent traffic light controlling algorithms using vehicular networks. *IEEE Transactions on Vehicular Technology*, 65(8):5887–5899.

[27] Yousef, K. M., Al-Karaki, M. N., and Shatnawi, A. M. (2010). Intelligent traffic light flow control system using wireless sensors networks. *J. Inf. Sci. Eng.*, 26(3):753–768.

[28] Zhang, S., Wu, G., Costeira, J. P., and Moura, J. M. (2017). Understanding traffic density from large-scale web camera data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5898–5907.