U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

PEEC - Urban traffic simulation and optimization

Advisor: Prof. João Pedro Pedroso

Author: Inês Façanha Gonçalves

Date: July 28, 2023

Faculty of Sciences of University of Porto

1 Introduction

1.1 Context

The global trend of transportation by vehicles is on the rise, especially in densely populated cities. As a result, there is a growing requirement to simulate and optimize traffic control in order to effectively manage this escalating demand [1].

1.2 Objectives/Motivation

The aim of this project is to study, from a conceptual point of view, how traffic lights could be improved for reducing wasted time.

Based on simulations of typical traffic patterns in a specific area, this project will generate recommendations by comparing the current operational state with an idealized scenario.

These recommendations aim to improve traffic flow and minimize time inefficiencies.

2 Theory

In the management and planning of intersections with traffic signals or without them, various performance indicators, such as queue lengths and delays, are crucial.

These indicators not only reflect the quality of service provided to drivers but also impact fuel consumption and air pollution levels.

Hence, they serve as significant measures to assess the overall effectiveness and environmental impact of intersection control and design [2]. In this project, the queue length was used as a performance indicator.

There are different types of traffic signal, in our case we have used the "Fixed Time Signals", with four different periods, red, amber, green and yellow. The red is the colour that indicates the driver to stop, the amber is the clearance time to clear all the vehicles, the green is the colour that tells the driver he can go and the yellow indicates that the driver should slow down. This is repeated in a loop in this order, that is named cycle time [3].

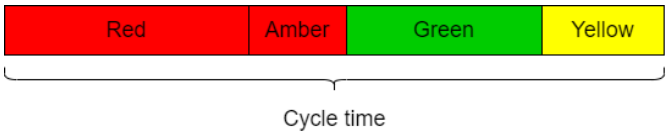

Figure 1: Traffic signal configuration.

3 Development

3.1 Tools and Materials

For the development of this task, it was used *PTV Vissim 2023* software [4]. Through the *COM API* it was possible to connect the software to *python*, version *3.11*.

3.2 Implementation

In this project we intended to analyze two different situations.

Firstly we wanted to analyse the traffic in a single intersection. We have analyzed the traffic behaviour with and without the presence of traffic signals.

Secondly, we have build a network with two intersections, in which the goal was to understand how the traffic on one intersection would affect the following one. This time we have only consider this situation using traffic signals.

In both situations we have observed the queue length value while increasing the vehicle volume by 200 veh/h, with a range from 200 to 2400.

3.2.1 Building the networks

In the following pictures it is possible to see how both networks were build. In the two networks the maximum vehicle velocity was 50 km/h.

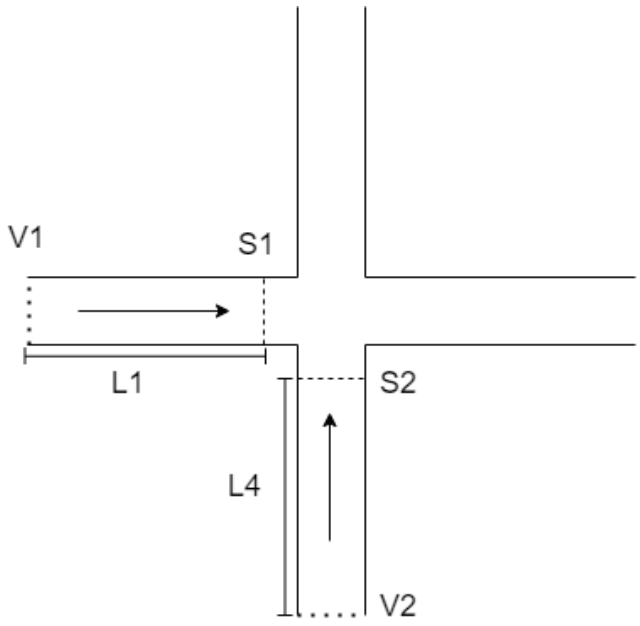

Figure 2: Network with one intersection.

The arrows represent the movement that the vehicles could make, in the case that was studied they could only move forward. The letters V, L and S represent the vehicle input, that is, the vehicle volume (veh/h), the length of the link, and the signal head, respectively.

On the first network both V1 and V2, were incremented by 200, ranging from 200 to 2400. The same happened for V1 in the second network, while V2 and V3 remained constant (200

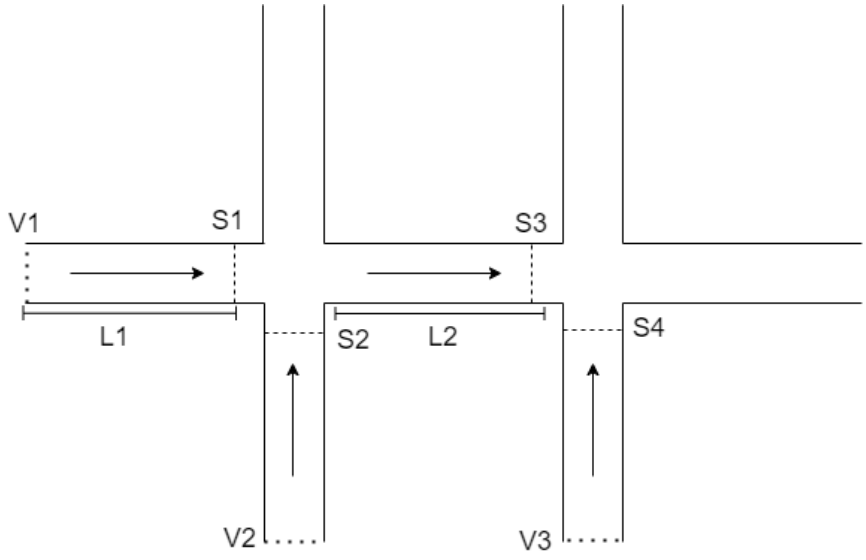

Figure 3: Network with two intersections.

veh/h). The process will be better explained in the next section.

On the tables below we can see the remaining data relative to the networks.

Name of the link	**Length of the link (m)**
L1	406
L4	402

Table 1: Links length for the network with one intersection.

Name of the link	**Length of the link (m)**
L1	201
L2	201

Table 2: Links length for the network with two intersections.

Since both networks are considered simple (the vehicles can only move forward), every signal head in both networks have the same configuration, which can be seen in the table below.

Cycle time	Red	Amber	Green	Yellow
100	50	2	45	3

Table 3: Signal heads configuration. The time unit is in seconds.

3.2.2 Code implementation

By using the *COM API*, we have built two python scripts to collect the queue length values in each network. The full scripts can be seen in the "Attachments" section.

In the first network the goal was to obtain some matrix like the one that can be seen below, in which we would obtain the queue length values for each link (1 and 4).

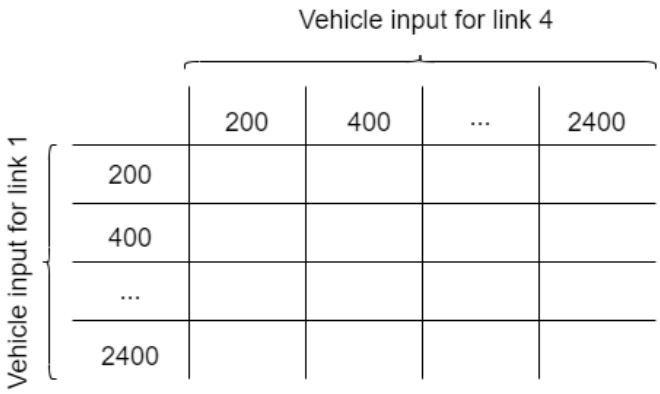

Figure 4: Matrix to be obtained.

To connect through the *COM API* we first have to go to the *PTV VISSIM* software and go to *Help* → *Register COM Server*. Afterwards, in the script we have to connect to *PTV VISSIM* and load the desired network.

```
import win32com.client as com
# Makes the conection to vissim
Vissim = com.Dispatch("Vissim.Vissim")
# Load a Network
Filename = ...path to the network
Vissim.LoadNet(Filename)
```

In order to get the desired data we have to set some parameters. As said before, the goal was to observe the queue length behavior of both links, with the increase of the volume. The range of the volume was from 200 to 2400 with a step of 200.

In order to set the volume value we have to access the link id, and attribute the volume.

```
# Setting Veh inputs and Volume
VI_number_1 = 1  # VI = Vehicle Input link 1
```

```
volume_1 = veh_input_1_list[i]  # vehicles per hour
Vissim.Net.VehicleInputs.ItemByKey(VI_number_1).SetAttValue('Volume(1)', volume_1)
```

We have run several simulations, with a default time duration of 3600 s. To do this we have to set the simulation to run continuously and give it a maximum duration.

```
# Setting simulation
End_of_simulation = 3600  # simulation second [s]
Vissim.Simulation.SetAttValue('SimPeriod', End_of_simulation)
Vissim.Simulation.RunContinuous()
```

To get the queue length values, we have again to give the link id we want to access.

```
# Getting data from link 1(queue length) -> counter 1
queue_counter_number_link1 = 1
queue_counter_1 = Vissim.Net.QueueCounters.ItemByKey(queue_counter_number_link1)
# Get the queue length value in meters
queue_length_1 = queue_counter_1.AttValue('QLen(Avg, Avg)')
```

We have run the script with the traffic signals activated and deactivated.

For the second network, we have done the same, but this time we wanted to know how L1 would influence L2, so the only vehicle input of interest was V1, the rest was kept constant.

It would not make sense not using traffic signals, since the "principal route" would have priority and the traffic would flow freely. Since, we wanted to observe how L1 would affect L2, we have only run the script one time (with the traffic signals activated).

4 Results

4.1 First intersection

4.1.1 Link 1

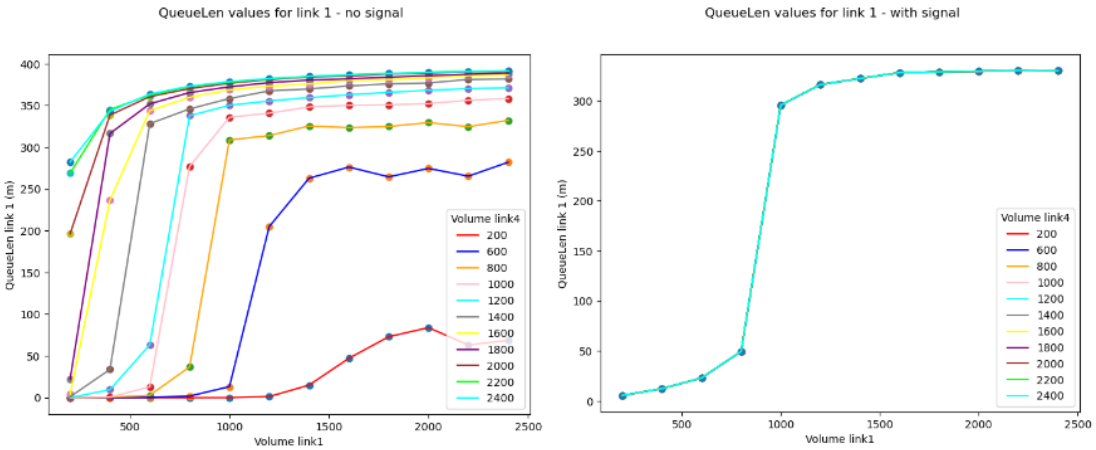

Figure 5: Results for link 1.

On the first picture we can see that with the increase of the volume, the queue length also increases until it starts converging to a maximum value of saturation. The higher the volume the quickest the queue length will reach that value. In this case we clearly see that the vehicle volume of the other link affects the traffic, since link 1 has no priority and the traffic signals are deactivated.

On the second one, again we can see that the higher the volume the higher the queue length until it reaches a maximum value. But this time, since the traffic signals are activated, the traffic flow will be independent from the other link, and depend only on the volume of its own.

We can also see that for low values of volume, we reach lower values of queue length by not using traffic signals.

4.1.2 Link 4

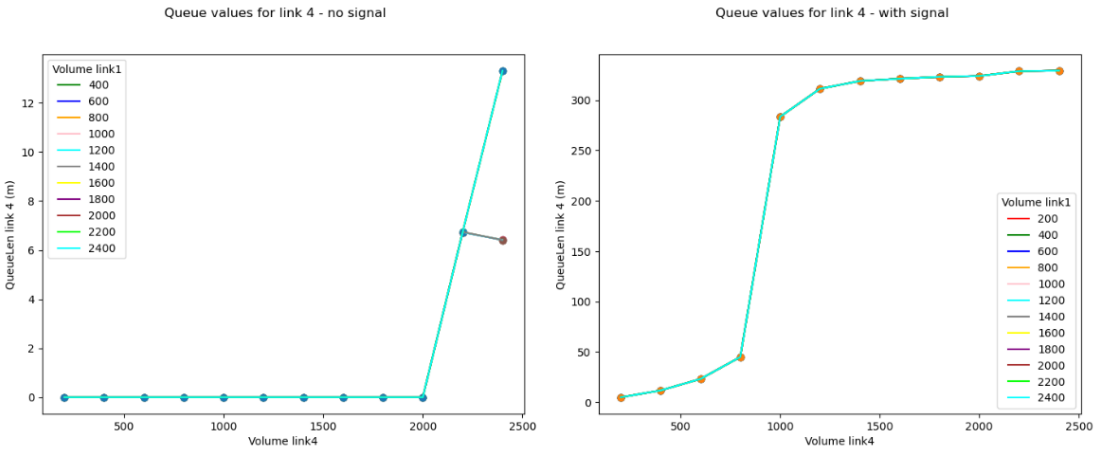

Figure 6: Results for link 4.

On both pictures we can, again, see that with the increase of the volume there is an increase of the queue. But this time, since link 4 has priority it will not depend on the traffic of link 1.

Also, for the second picture, because we are using signal heads, the traffic will only depend on its on volume, as we have already said, which is why we obtain the same values for both links by using traffic signals.

It is possible to see that we reach lower values of queue length when the traffic signals are deactivated, especially since this link is the one that has priority at the intersection.

4.2 Second intersection

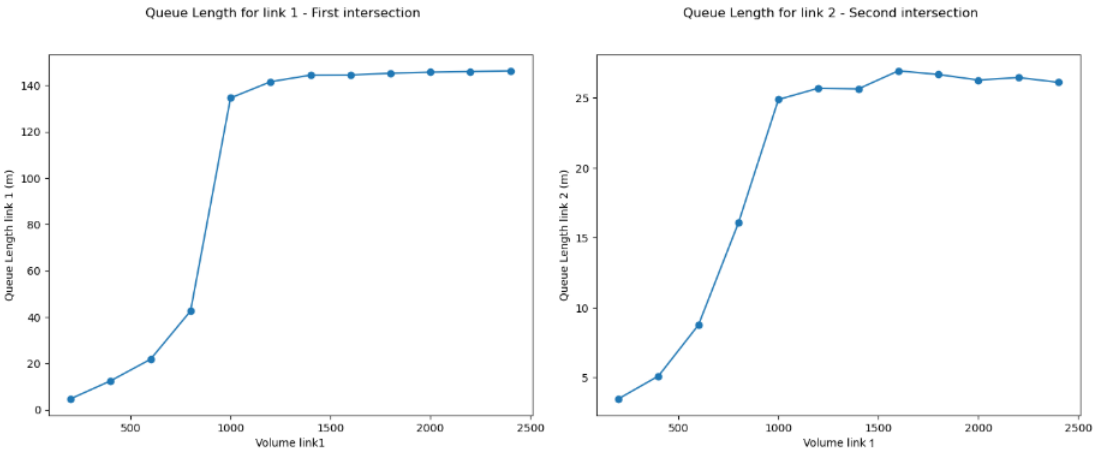

Figure 7: Results for link 1 and link 2.

With the increase on link's 1 queue length, the queue length on link 2 also increases. Both converge to a value, although link 2 keeps lower values of queue length, as traffic has already been reduce by link 1.

5 Conclusion and Future Work

The goal of this project is to optimize traffic by reducing wasted time.

We have focused on the queue length as a measure to analyze the traffic behaviour and we have understood that for lower volume of vehicle, it is better to have traffic signals deactivated. This could be interest for future applications.

We have also found out that the size of the queue before an intersection affects the following one, but this is something that was already expected.

In the future, further studies will be done: we will study a real life situation, find a corresponding optimal scenario and compare it with the current implementation.

References

[1] MA Wiering, J van Veenen, Jilles Vreeken, Arne Koopman, et al. Intelligent traffic light control. 2004.

[2] Dirk Heidemann. Queue length and delay distributions at traffic signals. *Transportation Research Part B: Methodological*, 28(5):377–389, 1994.

[3] Pranav Kumar Pal and Kamal Nabh Tripathi. Research on design of traffic signals at non-signalized intersections in lucknow city. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT)*, 11(04), April 2022.

[4] May 2023.

Attachments

A. Script of the first network

```
import win32com.client as com
import numpy as np

# Faz a conexão ao vissim
Vissim = com.Dispatch("Vissim.Vissim")
# Load a Network
Filename = r"C:\Users\nezfa\OneDrive\Ambiente de Trabalho\Estágio\CruzamentoUnilabs.inpx"
Vissim.LoadNet(Filename)

veh_input_1_list = [x for x in range(200, 2401, 200)]
veh_input_4_list = [x for x in range(200, 2401, 200)]
veh_input_1_print = []
veh_input_4_print = []
list_queue_length_1 = []
list_queue_length_4 = []
contador = 0

for j in range(len(veh_input_1_list)):
    for i in range(len(veh_input_1_list)):
        # Setting Veh inputs and Volume
        print(j)
        VI_number_1 = 1  # VI = Vehicle Input link 4
        VI_number_2 = 2  # VI = Vehicle Input link 1
        volume_1 = veh_input_1_list[j]  # vehicles per hour
        volume_4 = veh_input_4_list[i]  # vehicles per hour
        Vissim.Net.VehicleInputs.ItemByKey(VI_number_1).SetAttValue('Volume(1)', volume_4)
        Vissim.Net.VehicleInputs.ItemByKey(VI_number_2).SetAttValue('Volume(1)', volume_1)

        veh_input_1_print.append(volume_1)
        veh_input_4_print.append(volume_4)

        # Setting simulation
        End_of_simulation = 3600  # simulation second [s]
        Vissim.Simulation.SetAttValue('SimPeriod', End_of_simulation)
        Vissim.Simulation.RunContinuous()

        # Getting data from link 1(queue length) -> counter 2
        queue_counter_number_link1 = 2
        queue_counter_1 = Vissim.Net.QueueCounters.ItemByKey(queue_counter_number_link1)
        # Get the queue length value in meters
```

```
        queue_length_1 = queue_counter_1.AttValue('QLen(Avg, Avg)')


        contador = contador + 1

        if j == 0 and i == 0:
            queue_length_1 = round(queue_length_1, 2)
            list_queue_length_1.append(queue_length_1)
        else:
            somaDelays_1 = sum(list_queue_length_1)
            print("Soma: {0}".format(somaDelays_1))
            print("lista: {0}".format(list_queue_length_1))
            print("contador: {0}".format(contador))
            print("queue_length_1: {0}".format(queue_length_1))
            list_queue_length_1.append(round(((queue_length_1 * (contador)) -
            somaDelays_1), 2))
            print("lista depois: {0}".format(list_queue_length_1))

        # Getting data from link 4(queue length) -> counter 1
        queue_counter_number_link4 = 1
        queue_counter_4 = Vissim.Net.QueueCounters.ItemByKey(queue_counter_number_link4)
        # Get the queue length value in meters
        queue_length_4 = queue_counter_4.AttValue('QLen(Avg, Avg)')


        if j == 0 and i == 0:
            queue_length_4 = round(queue_length_4, 2)
            list_queue_length_4.append(queue_length_4)
        else:
            somaDelays_4 = sum(list_queue_length_4)
            list_queue_length_4.append(round((queue_length_4 * (contador)) -
            somaDelays_4, 2))




"""
print(list_queue_length_1)
print(list_queue_length_4)
"""
fileName = 'queueLength_no_signal.txt'
with open(fileName,'w') as textfile:
    print ("volume(1) volume(4) queueLength(1) queueLength(4)", file=textfile)
    for i in range(len(list_queue_length_4)):
        print("{0} {1} {2} {3}".format(veh_input_1_print[i], veh_input_4_print[i],
        list_queue_length_1[i], list_queue_length_4[i]), file=textfile)
```

B. Script of the second network

```
import win32com.client as com
# Makes the conection to vissim
Vissim = com.Dispatch("Vissim.Vissim")
# Load a Network
Filename = r"C:\Users\nezfa\OneDrive\Ambiente de Trabalho\Estágio
\CruzamentosConsecutivos.inpx"
Vissim.LoadNet(Filename)

veh_input_1_list = [x for x in range(200, 2401, 200)]
veh_input_1_print = []
list_queue_length_1 = []
list_queue_length_2 = []
contador = 0

for i in range(len(veh_input_1_list)):
    # Setting Veh inputs and Volume
    VI_number_1 = 1  # VI = Vehicle Input link 1
    volume_1 = veh_input_1_list[i]  # vehicles per hour
    Vissim.Net.VehicleInputs.ItemByKey(VI_number_1).SetAttValue('Volume(1)', volume_1)

    veh_input_1_print.append(volume_1)

    # Setting simulation
    End_of_simulation = 3600  # simulation second [s]
    Vissim.Simulation.SetAttValue('SimPeriod', End_of_simulation)
    Vissim.Simulation.RunContinuous()

    contador = contador + 1

    # Getting data from link 1(queue length) -> counter 1
    queue_counter_number_link1 = 1
    queue_counter_1 = Vissim.Net.QueueCounters.ItemByKey(queue_counter_number_link1)
    # Get the queue length value in meters
    queue_length_1 = queue_counter_1.AttValue('QLen(Avg, Avg)')

    # Getting data from link 2(queue length) -> counter 2
    queue_counter_number_link2 = 2
    queue_counter_2 = Vissim.Net.QueueCounters.ItemByKey(queue_counter_number_link2)
    # Get the queue length value in meters
    queue_length_2 = queue_counter_2.AttValue('QLen(Avg, Avg)')

    if i == 0:
```

```
            queue_length_1 = round(queue_length_1, 2)
            list_queue_length_1.append(queue_length_1)
            queue_length_2 = round(queue_length_2, 2)
            list_queue_length_2.append(queue_length_2)
        else:
            somaDelays_1 = sum(list_queue_length_1)
            list_queue_length_1.append(round(((queue_length_1 * (contador)) - somaDelays_1), 2))
            somaDelays_2 = sum(list_queue_length_2)
            list_queue_length_2.append(round(((queue_length_2 * (contador)) - somaDelays_2), 2))

        print("Volume 1: {0}".format(volume_1))
        print("Link1: {0}".format(list_queue_length_1[i]))
        print("Link2: {0}".format(list_queue_length_2[i]))


fileName = 'queueLength_two_intersections.txt'
with open(fileName,'w') as textfile:
    print ("volume(1) queueLength(1) queueLength(2)", file=textfile)
    for i in range(len(list_queue_length_1)):
        print("{0} {1} {2}".format(veh_input_1_print[i],
        list_queue_length_1[i], list_queue_length_2[i]), file=textfile)
```