DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2019

Reinforcement Learning Applied to Select Traffic Scheduling Method in Intersections

DAVID JOHANSSON

JOHAN VON HACHT

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Bachelor in Computer Science
Date: June 11, 2019
Supervisor: Erik Fransén
Examiner: Örjan Ekeberg
Swedish title: Förstärkande inlärning applicerad för val av scheduleringsmetod i korsningar
School of Electrical Engineering and Computer Science

Abstract

Effective scheduling of traffic is vital for a city to function optimally. For high-density traffic in urban areas, intersections and how they schedule traffic plays an integral part in preventing congestion. Current traffic light scheduling methods predominantly consist of using fixed time intervals to schedule traffic, a method not taking advantage of the technological leaps of recent years. With the unpredictable characteristic of traffic and urban population ever-expanding, conventional traffic scheduling becomes less effective due to them being non-adaptive. Therefore, the study sought out to investigate if a traffic scheduler utilising reinforcement learning could perform better than traditional traffic scheduling policies used today, more specifically fixed-interval scheduling. A solution involving a reinforcement agent choosing different predefined scheduling methods with varied characteristics was implemented. This implementation was successful in lowering the average waiting time of cars passing the intersection compared to fixed-interval scheduling. This was made possible by the agent regularly applying suitable scheduling method for the present traffic conditions. Reinforcement learning could, therefore, be a viable approach to scheduling traffic in intersections. However, the reinforcement agent had a limited overview of the current traffic environment at its disposal which could have consequences for the result.

Sammanfattning

Effektivt styrande av trafik utgör en väsentlig del av väl fungerande städer. I tätbefolkade områden med hög trafikdensitet spelar schedulering av korsningar en viktig roll i form av att förhindra långa köer. I dagsläget används till stor del trafikljus som styrs av förbestämda tidsintervall, en metod som inte utnyttjar de teknologiska framsteg som gjorts under de senaste åren. Trafikens oförutsägbarhet samt den ständigt ökande populationsmängden ställer krav på allt mer effektivare trafikljus, trafikljus som kan anpassa sig utefter den rådande trafiksituationen. Därmed undersöker denna rapport huruvida trafikljus som använder sig av förstärkande inlärning kan prestera bättre än konventionella metoder, mer specifikt schedulering med förbestämda tidsintervall. En lösning implementerades som utnyttjar förstärkande inlärning i mån om att välja mellan fem olika trafikstyrningsmetoder med utmärkande egenskaper. Metoden lyckades förbättra den genomsnittliga väntetiden för bilarna som passerade korsningen jämfört med väntetiden som förbestämda tidsintervall åstadkom. Detta genom att regelbundet välja den metod som presterade bra för den givna trafiksituationen. Förstärkande inlärningen kan därmed ses som en lämplig metod för att styra trafiken i korsningar. Lösningen hade dock en begränsad överblick av omgivningen vilket skulle kunna påverka resultatet.

Contents

Chapter 1

Introduction

Good traffic flow serves as a vital component for a city to function properly. Traffic enables resources to be transported effectively and increases the overall quality of life for the city's inhabitants. Achieving good traffic flow is largely dependant on traffic lights being able to control vehicular movement efficiently through intersections without causing congestion. Intersections today usually control traffic through fixed time intervals, a control method that gives each lane of the intersection equal amount of time for cars to pass [8]. The prevalence of random aspects such as traffic collisions and the exponential population growth in urban areas are examples of factors that are difficult for the aforementioned traffic-control techniques to accommodate for.

With a society considerably dependent on self-driving cars possibly on the horizon, new avenues for traffic control opens up. One such solution, made possible by modern technological advancements, is letting a computer generate a traffic scheduling approach on its own through means of machine learning. This study will explore the possibility of improving upon conventional traffic control by utilising machine learning, more specifically reinforcement learning. Reinforcement learning is a method where a system is trained to attain a specific goal by taking suitable actions in a particular environment [10]. Autonomous cars being able to communicate with traffic lights facilitate the prospect of implementing a traffic controller using reinforcement learning. This is because the traffic controller can be given the ability to control the movement of each car throughout the intersection.

1.1 Purpose

Evaluating the viability of reinforcement learning is important because of the potential benefits that the application could have. Firstly, more efficient traffic control results in cars spending less time on the roads which subsequently also results in a reduced amount of pollution. In addition to the environmental benefits, the application of reinforcement learning could potentially result in a reduced economic cost by decreasing the number of congestions. According to Cookson [2], congestion cost the US approximately $305 billion in 2017.

1.2 Problem Statement

Can a traffic scheduler utilising reinforcement learning reduce the average waiting time in a 4-way intersection compared to a traditional method in the form of fixed-time scheduling?

1.3 Scope

There are many different intersections present in current traffic networks with diverse characteristics. This study will focus on a 4-way intersection with two lanes in each direction. To simplify the problem case further, pedestrians and differences in car sizes will not be taken into consideration when modelling the intersection. The cars will not be represented as physical objects but rather as logical, i.e. physical properties such as acceleration and size of the vehicle will not be accounted for.

1.4 Thesis Outline

In the subsequent chapter, the reader will be acquainted with the traffic characteristics that have been taken into consideration when modelling a simulation of the intersection. The reader will also be presented with an introduction to reinforcement learning. Furthermore,

previous studies revolving around traffic scheduling will be described. Chapter three will outline the methodology used to model the simulation and the algorithm used to construct the reinforcement agent. In the fourth chapter, the results obtained are laid out and then discussed. Finally, a conclusion consisting of the main takeaways garnered from the discussions are presented in chapter six.

Chapter 2

Background

In order to implement a dynamic traffic scheduler, general information about traffic characteristics, as well as how traffic scheduling is carried out in current intersections needs to be known. Therefore, this chapter will present statistics of current traffic patterns and the general types of traffic scheduling methods utilised today. This is followed by an explanation of what reinforcement learning is as well as different types of implementations. The chapter will end by presenting related literature that ties into the work that has been carried out in this study.

2.1 Traffic Characteristics

The traffic volume imposed on real-life intersections varies throughout the day. This becomes apparent through the findings of Han, Wang, and Shaw [3] which showed that the average traffic that passes through an intersection in a day is characterised by two significant peaks during work hours. The first peak occurs between 8 am and 12 am as most drivers are travelling to work. This peak stays roughly the same during weekdays but is less pronounced during weekends. The same goes for the second peak which takes place between 2 pm and 6 pm as travellers are primarily driving home. The complete volumetric data can be seen in figure 2.1. A similar bimodal characteristic of traffic in intersections is referenced by Leduc [6] in a working paper published by the Joint Research Centre.

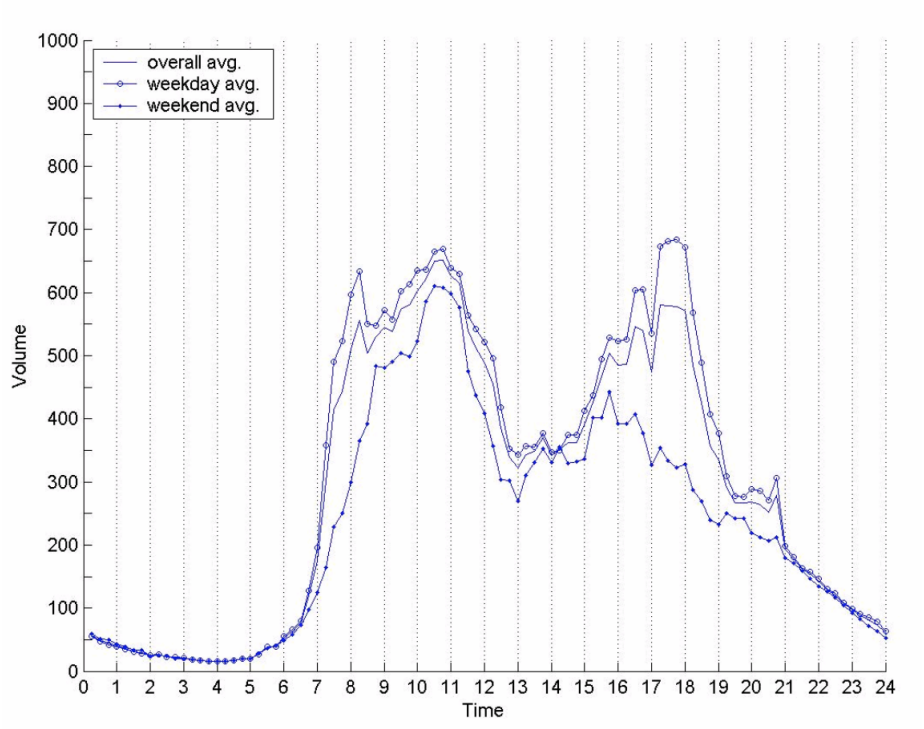

Figure 2.1: Volume of traffic at an intersection monitored by Han, Wang, and Shaw [3].

2.2 Current Traffic Control

Most commonly utilised types of traffic control in current intersections are either non-adaptive or adaptive. Non-adaptive alternatives primarily involve having traffic lights being controlled by fixed-time intervals. These intervals are based solely on historical traffic data and, therefore, stay the same regardless of traffic conditions which potentially results in congestion [11].

Similar to the non-adaptive alternative, adaptive traffic control is also generally based on historical data. However, the intersection's signal plan can be changed as a result of them being monitored or by using vehicle actuation [8]. Examples of factors that could result in the fixed traffic intervals being modified include special events and weather conditions. These are events that non-adaptive traffic control cannot take into account. Especially considering that traffic engineers

will often use very neutral traffic data sets to base traffic light intervals on, making sure to primarily accommodate for the most common conditions [11].

2.3 Reinforcement Learning

Reinforcement learning is an area in machine learning where the computer aims to learn which actions to perform in certain situations without being told explicitly what to do. Since the algorithm has no specific instructions to follow it has to discover the best actions to take, i.e. the ones who give the best outcome by interacting with the environment [10].

2.3.1 Keywords

The following section will define some essential keywords relating to reinforcement learning.

State
A state is a condition of the environment. It describes what the current environment looks like [10].

Agent
An agent is an entity that is integrated within an environment and has the ability to perform actions in order to change the state of the environment [10].

Reward
A reward in the context of reinforcement learning is a measurement of how good the outcome of a certain action performed by the agent was. The reward can either be negative or positive but should allow the reinforcement learning algorithm to determine what candidates as a good or bad action in the given state [10].

Discount Factor
The discount factor, usually denoted as γ, decides how much future rewards are worth compared to immediate rewards. It is a number between zero and one where a low discount factor lessens the importance of future rewards while a high factor increases it [12].

Learning Rate
Learning rate, commonly denoted as α, dictates how much the agent should learn from the actions taken. A low learning rate means that the agent should depend more on prior actions instead of recent information. A higher learning rate, on the other hand, will result in the agent mostly considering recent information when making decisions. In a stochastic environment, lower learning rate values are preferred while a higher learning rate is better suited for deterministic scenarios [12].

Policy
A policy defines how the agent should behave by mapping the perceived state of the environment to the appropriate action to take [10].

2.3.2 Exploration and Exploitation

One of the most prominent challenges in reinforcement learning is handling the trade-off between exploration and exploitation. To obtain reward, the reinforcement learning agent needs to *exploit* actions that it has previously found to be effective in regards to producing reward. However, to find the most reward-inducing actions, the agent needs to *explore* new actions that it has not yet selected. Exclusively pursuing one or the other will most likely result in poor performance. For instance, only exploring corresponds to always choosing actions randomly. This might result in the optimal action being taken in certain situations but defeats the purpose of implementing a reinforcement learning agent in the first place [10].

Both exploration and exploitation can be utilised by implementing an action selection policy in the form of an ϵ-greedy policy. The ϵ-greedy policy will most commonly choose the action that will result in the

highest estimated reward. Moreover, the policy is able to choose a random action with the probability of ϵ which ensures that more beneficial behaviours can be found by the agent. With all of this in mind, the policy can both exploit previous actions found to be profitable and explore new possible behaviours [10].

2.3.3 Q-learning

Q-learning is a model-free reinforcement learning algorithm which means that it does not need to have a model of the environment in order to learn. The goal of the Q-learning algorithm is to find the best action to take under certain circumstances. The algorithm works by defining an agent, states *S* and actions *A*. The agent can transition from and to different states by performing an action which gives it an outcome-based reward. The algorithm strives to find the set of actions from each state that nets the highest discounted future reward *R*, a reward calculated by equation 2.1 where *r* is the reward given at time *t* [12].

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + ... + \gamma^{n-t} r_n \tag{2.1}$$

The reward (Q-value) of a state and action is calculated with the Q-function in equation 2.2 [10].

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old Q value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot [\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_{\alpha}(Q(s_{t+1}, a))}_{\text{max future Q-value/reward}} - \underbrace{Q(s_t, a_t)}_{\text{old Q-value}}] \tag{2.2}$$

One way to store the Q-values is in a table of size $S \times A$ where S is the number of states and A the number of actions. This method, however, can require vast amounts of computer memory since problems often have many or even an infinite amount of states. A solution to this problem is to utilise a neural network in conjunction with Q-learning, often called Deep Q-Network, a solution outside the scope of this study.

Double Q-learning

In certain stochastic environments, the standard Q-learning algorithm runs the risk of overestimating action values according to Hasselt [4]. Overestimation occurs since Q-learning uses the same Q-values to decide which actions to take as well as to predict the new Q-values. As a result, the algorithm can favourably estimate actions that are tried often even though other actions might result in higher reward. Hasselt continues by proposing a solution to this problem in the form of *Double Q-learning*. By utilising two estimators, i.e. two Q-value functions as seen in equation 2.3 and 2.4, the algorithm no longer suffers from overestimation.

$$Q^A(s_t, a_t) \leftarrow Q^A(s_t, a_t) + \alpha \cdot [r_{t+1} + \gamma \cdot (Q^B(s_{t+1}, \arg\max Q^A(s_{t+1}, a_t))) - Q^A(s_t, a_t)] \quad (2.3)$$

$$Q^B(s_t, a_t) \leftarrow Q^B(s_t, a_t) + \alpha \cdot [r_{t+1} + \gamma \cdot (Q^A(s_{t+1}, \arg\max Q^B(s_{t+1}, a_t))) - Q^B(s_t, a_t)] \quad (2.4)$$

On the other hand, Double Q-learning can suffer from underestimation. Despite this Double Q-learning showed faster learning compared to Q-learning in certain stochastic environments [4].

2.4 Related Work

Research regarding computer-assisted optimisation of traffic control dates back several decades. Earlier research presented, for instance, solutions utilising linear programming to determine optimal intervals for traffic light controllers. However, these solutions were modelled after limited information in regards to larger traffic systems with several interconnected intersections making the technology difficult to apply on a wider scale [7].

With road networks becoming more complex, researchers needed to find a more dynamic approach to optimising traffic control. Between 1997 and 2010, solutions using reinforcement learning primarily consisted of using model-free Q-learning. Due to the limitations of the technology at the time, a relatively small state space had to be used

limited to, for instance, the number of vehicles [7]. A model-based alternative approach is described by Wiering [13] and presents a solution based on a transition model that maps the agent states to long term rewards. This makes it possible to speed up the necessary learning time significantly and thus allowing for more parameters to be included in the state space.

To make reinforcement learning more applicable in a broader context, research has also proposed solutions in regards to having multiple intelligent agents coordinating with each other. This makes it possible to dictate traffic in complex traffic systems consisting of several intersections [1].

Instead of controlling the traffic flow by manually selecting lanes or cars, reinforcement learning can also be applied to select the best traffic scheduling method for the current traffic condition. This approach has been tested by Jansson and Uggla Lingvall [5] but applied to elevator control where the metric of performance was squared waiting time. The results showed an improvement in overall waiting time when the reinforcement agent was applied to larger elevator environments.

Chapter 3

Method

The following section will outline the method used to answer the problem statement. Firstly, the method will consist of creating a simulation serving the purpose of recreating the conditions of an intersection. The simulator will subsequently be used to test two different approaches to traffic control in the form of fixed-time intervals and intelligent traffic scheduling using reinforcement learning.

3.1 Simulation

A simulator that recreates the traffic conditions in a 4-way intersection was constructed with its intended purpose being to facilitate the training of the reinforcement learning agent. It was programmed using Python due to the authors being familiar with it and the fact that there are plenty of resources relating to machine learning written in the language. The simulator was logical meaning that entities, i.e. cars that entered the intersection, did not have physical attributes like length or weight. Instead, they were represented as logical occupiers, a characteristic that will be described in further detail in section 3.1.1.

3.1.1 Intersection Model

The modelled intersection had eight lanes in total with two coming from each direction. The two lanes in each direction allowed the cars

to turn into every other lane. These lanes were represented as queues adhering to the FIFO (first in, first out) policy, meaning that the first car that entered the queue would be the first one to leave. For clarity, the different lanes will be referred to as their cardinal directions moving forward. An illustration of the intersection is depicted in figure 3.1.

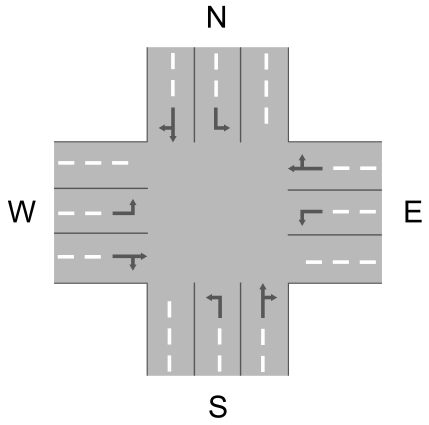

Figure 3.1: Illustration of the 4-way intersection that the simulation was based on.

To prevent the occurrence of collisions between cars, different lanes were not able to empty their queues in close temporal proximity. For instance, this is the case for the northern and the eastern lanes who could not empty their queues going either straight or right at the same time. To make sure no collisions happened, the simulation did not allow switching between lanes without waiting for a certain period. The chosen value for this was 7 seconds, a time interval determined to ensure that the last car had enough time to cross the intersection. The simulation also did not allow cars to drive directly after each other as there was a delay of two seconds between each car leaving from the same lane. This delay provided the cars with enough time to leave the lane without the trailing car potentially colliding from behind.

3.1.2 Simulation Time

The simulation was based on a fixed time model, meaning that the simulation had an internal timer that was updated by a fixed interval. All events, such as cars leaving their queues, that occurred in between

the intervals were treated as if they took place simultaneously at the end of the interval. The length of each timestep was 200ms which amounts to 18 000 timesteps per hour. This timestep length was chosen as it did not interfere with any aspects of the simulation. Having a too large timestep could lead to constraints with the selection of delays between cars. For example, having a timestep of 2 seconds would only allow delays to be a multiple of 2. Additionally, it is large enough to not make the simulation take too much time to execute. In total, the simulation spanned 24 hours which resulted in it taking 432 000 timesteps to execute.

3.1.3 Traffic Density Model

The arrival of cars was modelled after a probability function. This function took the time of day as a parameter and returned the probability of a car arriving to the intersection at that time. The probability function aimed to simulate realistic traffic by being based on the traffic data presented in figure 2.1. The most noteworthy characteristic of the data is its bimodal appearance as it depicts two clear peaks between the hours 7-12 and 15-19. These were the intervals with the highest probability of a car spawning.

Two different traffic density scenarios were simulated to understand how the reinforcement learning implementation performed in different conditions. The first scenario emulated even traffic with each lane receiving the same amount of vehicles. The second scenario differed in the sense that the north and south lanes were populated with twice the amount of traffic compared to east and west.

3.2 Reinforcement Learning Implementation

Similar to Jansson and Uggla Lingvall [5] mentioned in section 2.4, the solution was based around the idea of having a reinforcement learning agent with the ability to switch between different predefined traffic scheduling methods. These scheduling methods are defined in section 3.2.1. The reinforcement learning implementation utilised double Q-learning, described in section 2.3.3 where the intent of the al-

gorithm was to choose the scheduling method that would yield the lowest average squared waiting time depending on the magnitude of the incoming traffic. Double Q-learning was used instead of the standard Q-learning implementation since the simulated environment was stochastic. Another benefit of Double Q-learning is that it according to Hasselt [4] requires less time to train.

3.2.1 Scheduling Methods

The following methods were used by the Q-learning algorithm to schedule traffic.

Fixed Interval

Fixed interval scheduling is reminiscent of the conventional scheduling used in current intersections as it gives equal time for every lane to open up their queues. It starts by allowing cars going straight or right from the north and south to progress for 40 seconds. The cars going left from the two directions are then able to pass for 20 seconds. This amounts to one minute of total elapsed time where the north and south lanes are being emptied. The process is then repeated for the west and east lanes.

Fixed Interval, Prioritise Direction

The fixed interval, prioritise direction method is similar to fixed interval scheduling as it starts by opening up the north and south lane and then the west and east lanes. However, this scheduling method does not spend equal time before each switch. It prioritises lanes such as the north and south by spending more time opening up these lanes. For instance, if the north and south lanes should be prioritised, it would spend 80 seconds opening these and subsequently only spend 40 seconds on the west and east lanes. This scheduling method could prove to be useful if traffic is heavier from a certain direction.

First In, First Out

The first in, first out (FIFO) scheduling method prioritises the lane whose first car in the queue has the highest waiting time. For instance, if the first car in each lane has the following waiting times {10, 15, 5, 20}, it will open up the lane with its first car having waited for 20 timesteps. This scheduling method has the potential to perform well in a low traffic environment where the fixed interval scheduler would potentially have cars wait even though the intersection is empty elsewhere.

Longest Queue First

The longest queue first (LQF) scheduling method select the lane with the largest amount of cars waiting. This method could prevent long queues from building up which otherwise would lead to long waiting times. However, if the lanes have similar amounts of cars waiting, this scheduling method could potentially cause problems by being forced to switch between different lanes often. A switching process which takes time and prevents any cars from getting through.

3.2.2 Q-learning Algorithm

The Q-learning algorithm was based on the implementation by Hasselt [4] where actions were selected with an epsilon-greedy strategy. The detailed implementation can be seen in figure 3.2. The decision to determine which table to update, i.e. A or B, was decided at random. The simulation allowed the agent to change scheduling method every 600 timesteps and was rewarded based on how well traffic was scheduled during that time.

Reward

The agent was rewarded the negative average squared waiting time for each time period and action. The decision to use waiting time as the measurement of success for the agent was due to the fact that it is

Algorithm 1: Double Q-Learning

$Q^A(1..S, 1..A) \leftarrow 0$
$Q^B(1..S, 1..A) \leftarrow 0$
for 0 **to** $episodes$ **do**
  $reset\ environment$
  $done \leftarrow False$
  while $\neg done$ **do**
    `/* Select action using` ϵ`-greedy */`
    $action \leftarrow \epsilon$-greedy_selection()
    `/* Perform the action */`
    $next\_state, reward, done \leftarrow$ environment.step($action$)
    `/* Randomly update either` Q^A `or` Q^B `*/`
    if *update* Q^A **then**
      $old\_val \leftarrow Q^A(state, action)$
      $next\_best\_action \leftarrow \arg\max Q^A(next\_state, action)$
      $Q^A(state, action) \leftarrow old\_val + \alpha \cdot (reward + \gamma \cdot (Q^B(next\_state, next\_best\_action) - old\_val)$
    else if *update* Q^B **then**
      $old\_val \leftarrow Q^B(state, action)$
      $next\_best\_action \leftarrow \arg\max Q^B(next\_state, action)$
      $Q^B(state, action) \leftarrow old\_val + \alpha \cdot (reward + \gamma \cdot (Q^A(next\_state, next\_best\_action) - old\_val)$
  end
end

Figure 3.2: Pseudo code of Q-learning implementation.

the metric the study seeks to improve in the problem statement. Additionally, rewards based on metrics such as cars passed can lead to the agent never switching lanes since this allows more cars to pass at the cost of waiting time. The reason for making the reward negative was because the Q-learning algorithm tries to maximise the reward, which with a negative waiting time as the reward is equal to minimising the waiting time. The squared waiting time was used to emphasise that leaving a car waiting for very long while letting others pass is an undesired behaviour. This metric is also mentioned by Sutton and Barto [9] as a commonly used method to keep waiting times low while also serving all cars fairly.

A state in the simulation was a number between 0 and 624 constructed from the current traffic condition. Each lane could be in five different traffic states: no traffic, low traffic, medium traffic, high traffic and very high traffic. Each traffic condition had a predetermined value which then could be converted into a number and combined with the other lanes traffic conditions into a state.

Hyperparameters

Hyperparameters were set based on experimentation as well as from observing what these values usually are set to.

ϵ-$start$: 1

ϵ-$decay$: 0.99995

ϵ-min: 0.05

The epsilon value was decreased during learning, thus allowing the agent to explore the environment in the beginning. Epsilon was decayed each iteration resulting in the epsilon minimum being reached after around 60 episodes. A minimum epsilon of 0.05 allowed the Q-learning algorithm to explore on average once every hour in simulation time.

α (learning rate): 0.2

γ (discount factor): 0.85

A low alpha value was chosen since the simulated environment was non-deterministic. The gamma value was chosen by looking towards similar studies where the discount factor is often arbitrarily set close to 0.9. For instance, a discount factor of 0.95 was used for all of the experiments in the Double Q-learning study [4].

Chapter 4

Results

Table 4.1 displays the squared average waiting time over the span of a 24-hour simulation for the two traffic scenarios tested. In an environment where traffic is added stochastically in regards to the bimodal function described in section 2.1, the Q-learning agent accounts for the lowest squared waiting time. The agent is subsequently followed by the three fixed-time schedulers and lastly FIFO and LQF.

	FIFO	LQF	Fixed	Fixed NS	Fixed WE	Q-Learning
Even traffic	10 834.98s	17 914.00s	2132.79s	2144.33s	2158.13s	1806.88s
Heavier north/south	143 041.94s	239 114.03s	2288.89s	2171.29s	88 952.23s	2022.12s

Table 4.1: Average squared waiting time for each scheduling method during 24 hour simulation.

4.1 Even Traffic Scenario

The following section presents results from the even traffic scenario. In this scenario, the amount of incoming traffic was the same for all directions.

Figure 4.1 shows the agents performance during training. The average waiting time for the cars traversing the traffic light starts at approximately 300 seconds and as the agents learn throughout the learning episodes, it is reduced to 180 seconds. The reduction in waiting time

corresponds to a 40% improvement compared to selecting a scheduling method at random. The agent does not improve its performance past 100 episodes.

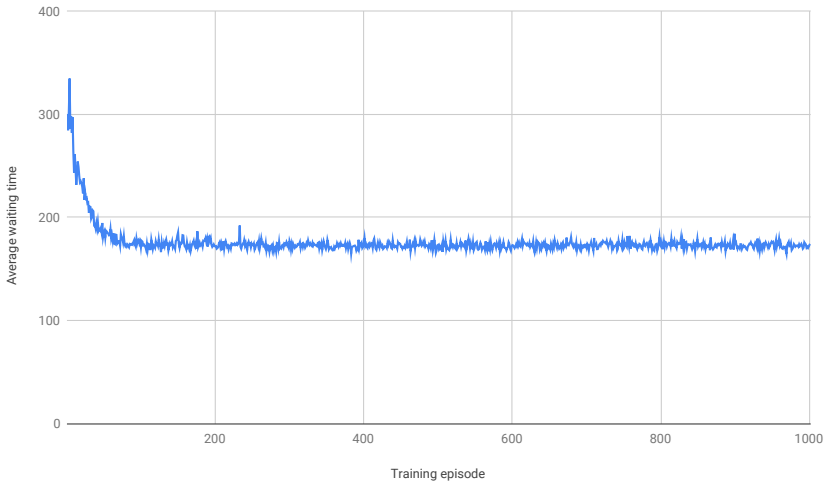

Figure 4.1: Total average waiting time over training episodes.

Figure 4.2 presents the hourly average waiting time for each scheduling method and Q-learning agent during the course of 24 hours. The result shows that different scheduling methods perform differently depending on the traffic density. With lower traffic, FIFO and LQF outperform the fixed interval methods significantly. However, during more demanding periods, the performance of FIFO and LQF deteriorates considerably. This is illustrated in the figure by using a logarithmic scale for the vertical axis.

A lowered performance is not noticeable for the fixed interval methods as their performance neither worsens nor improves during the course of the simulation. Therefore, the fixed interval methods vastly outperform FIFO and LQF methods as the traffic peaks. The Q-learning agents average waiting time stays close to the best scheduling method at any given time.

Figure 4.3 shows which actions were taken by the reinforcement agent during the 24-hour simulation. The figure shows that FIFO was used when there was low traffic while LQF was used during hours just before and after peak hours. During periods of high traffic, the fixed schedulers were favoured.

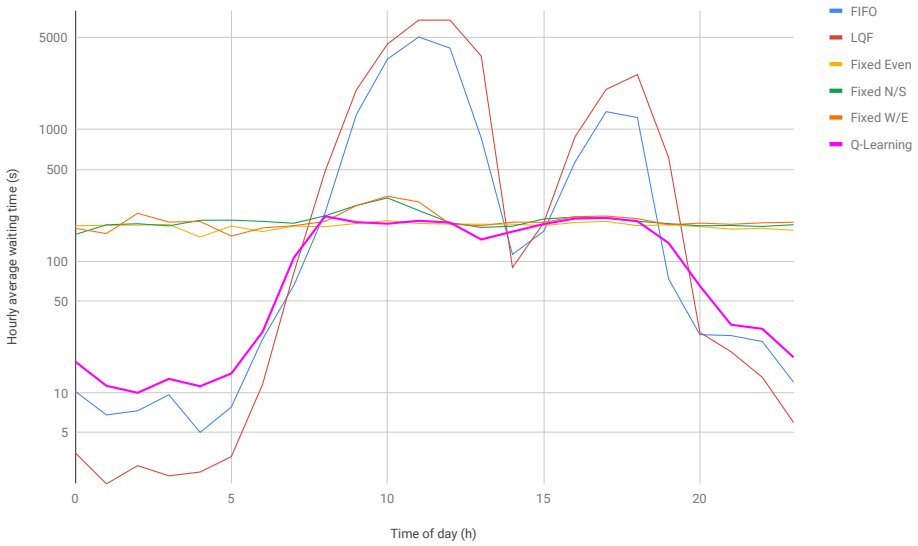

Figure 4.2: Average waiting time per hour for the scheduling methods.

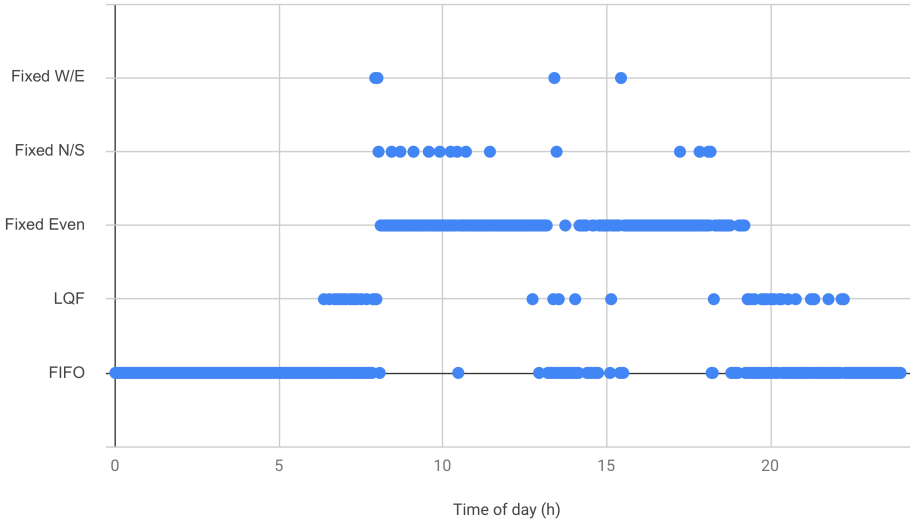

Figure 4.3: Action selected throughout the day.

4.2 Heavy North and South Traffic Scenario

The following section presents results from the second tested scenario with heavy traffic coming from north and south. In this scenario traffic from these directions were two times as high compared to the west and east. Due to traffic being twice as high from one lane, the overall traffic in this scenario was higher than the even traffic scenario.

Figure 4.4 shows the hourly average waiting time for each of the scheduling methods. FIFO and LQF perform well at the beginning of the simulation but is quickly overcrowded by cars. LQF is not able to schedule cars fast enough leading to the average waiting time never returning to the low values seen at the start of the simulation. The fixed interval scheduler and the fixed prioritised schedulers focusing on north and south performs well in high traffic. Even though traffic is two times as high from north and south, they do not differ much in performance. The fixed interval scheduler prioritising west and east, on the other hand, performs significantly worse and much like LQF is overcrowded by cars even at the end of the simulation.

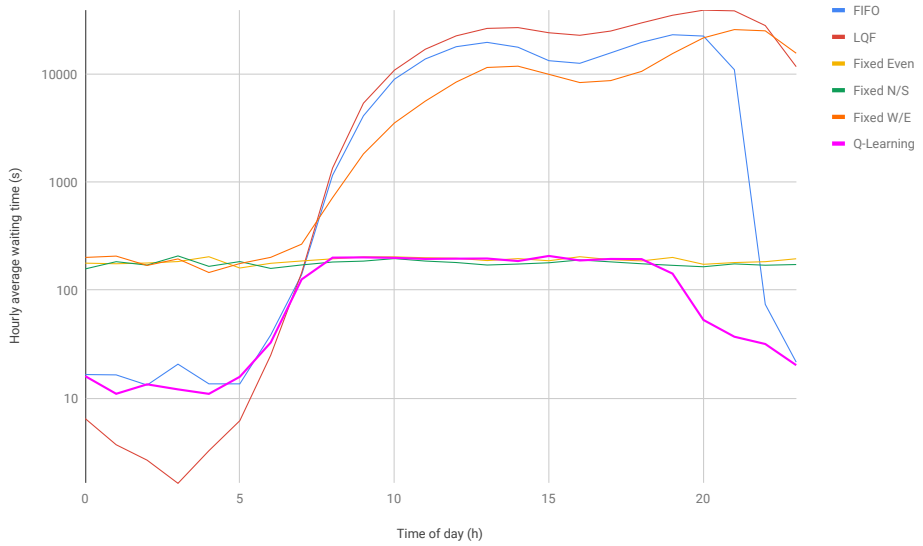

Figure 4.4: Average waiting time per hour for the scheduling methods.

Figure 4.5 displays the actions selected throughout the simulated day. The agent selects FIFO when traffic is low and the even fixed interval scheduler is the predominant method used when traffic is high. The fixed scheduler that prioritises north and south is selected a few times during peak hours. Similar to the other traffic scenario, LQF is used just before and after the traffic peaks seen in figure 2.1.

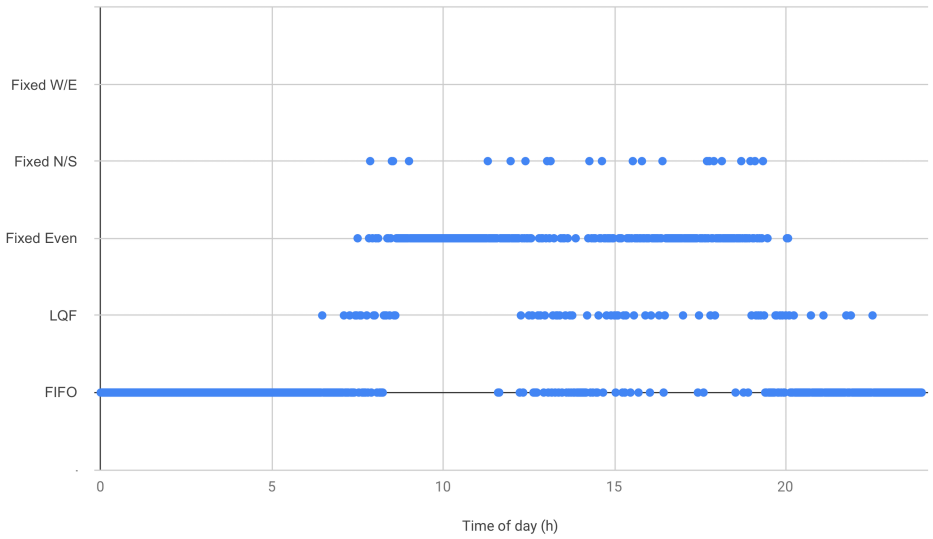

Figure 4.5: Action selected throughout the day.

Chapter 5

Discussion

According to the results gathered from the simulation, the reinforcement agent improves upon the performance of fixed interval scheduling in both even and heavy traffic as seen in table 4.1. This is most noticeable during low traffic hours where the Q-learning agent learns to utilise either FIFO or LQF, methods that heavily improves upon the performance of the fixed time scheduler during this traffic condition. Moreover, the Q-learning agent mostly adheres to the fixed interval methods when the traffic volume is at its peak which prevents large traffic congestion otherwise caused by the FIFO and LQF methods. This results in the reinforcement agent performing the best overall as it can adapt to both low and high-density traffic.

Judging by figure 4.2, there is no reason to use either the FIFO or LQF methods during dense traffic conditions. However, this might not paint the whole picture as they still might be used for this sort of traffic. This is not conveyed by the graph as it presents the average hourly waiting time. However, by looking at figure 4.5, it can be noted that FIFO and LQF get selected sporadically during the two high-density intervals. The reason for this could be that LQF, for example, is useful during circumstances where large congestion has been built up in a specific lane.

In figure 4.1, it is apparent that no significant improvements are being made in terms of total average waiting time after about 100 episodes of training. The average waiting time is especially large in the beginning as the epsilon value has not yet fully decayed and scheduling

methods are largely chosen at random. However, after 40 episodes, the epsilon value will have reached its minimum resulting in future scheduling methods mostly being determined by referring to the Q-tables. When the average waiting time starts to stagnate after 100 episodes, this could be caused by the fact that the Q-table is fairly small in regards to the number of states, hence, a limited amount of training is necessary.

In figure 4.2 it is shown that the reinforcement learning agent at some occasions performs better than all of the other scheduling methods, note hour 13. In theory, this should not be possible since the agent is not supposed to be able to perform better than the best scheduling method it can use. One potential reason for this is that two scheduling methods work well in conjunction with each other and therefore perform better than solely using one or the other. Strengthening this argument is the fact that two different methods were used during hour 13 as seen in 4.3 and 4.2. Another reason could be that the schedulers are not tested on identical traffic conditions. They are simulated with the same stochastic probability of cars spawning thus resulting in possible discrepancies in traffic conditions between simulation runs.

In the heavy south and north traffic scenario, the Q-learning agent learned to apply FIFO and LQF during low traffic hours. During high traffic, the agent was expected to prefer the fixed scheduler that prioritises the north and south lanes due to traffic being twice as high from these directions. However, this does not seem to be the case as the agent mostly used the even fixed scheduler. One of the reasons for this behaviour could be that the difference between the average waiting time, i.e. the reward, between the even fixed scheduler and the one that prioritises north and south was too small, see figure 4.4. This indicates that the agent has a limit to how well it can learn the optimal policy in scenarios where the difference in reward is narrow.

5.1 Reliability and Safety

An ideal reinforcement learning agent would always be able to choose the optimal scheduling method depending on the traffic characteristics. Judging by figure 4.2, this is not always the case as the Q-learning agent differs in performance from the optimal scheduling method during some intervals. This is most palpable during the initial part of the simulation as LQF outperforms the Q-learning agent. The reason behind this could potentially be that the rewards have not been optimally implemented leading to the result potentiality becoming less reliable.

Another explanation that could also impair the reliability of the result is that not enough training of the reinforcement agent has been carried out. However, considering the result displayed in figure 4.1, this is unlikely as the overall performance of the agent only marginally improves during the course of the last 900 episodes of training.

One factor that can affect the degree of applicability for the reinforcement learning solution is the fact that the simulation is logical. This results in several physical attributes of the cars not being taken into consideration when training the agent. This could potentially produce an agent not capable of scheduling traffic in real life scenarios while ensuring that safety is being preserved. This is by virtue of, for instance, cars of different lengths being given the same amount of time to cross the intersection in the simulation. This could subsequently make it difficult for human drivers to keep a safe distance between adjacent cars if they are driving a significantly larger vehicle. However, considering that the simulation is created with a future scenario in mind where all cars are fully autonomous, creating a logical simulation can be seen as sufficient. With autonomous cars, appropriate distance control between vehicles can be assured by the traffic scheduler itself as it can control the vehicles.

The solution presented in the study is reminiscent of vehicle actuation described in 2.2 considering that they both receive information about the presence of cars in each lane. Additionally, vehicle actuated traffic intersections use a policy closely resembling FIFO when traffic is very low, just like the implemented reinforcement learning agent. However, the agent has a broader view of the intersection as it is aware of how many cars are present in each lane, not only if there is a car at

the front of the lane. Therefore, it can prioritise certain directions that are more congested making it more powerful in comparison to vehicle actuation. Furthermore, the fact that vehicle actuation is applicable for current traffic control could signify that the same could be said for the reinforcement learning agent.

5.2 Future Work

The reinforcement learning algorithm manages to combine scheduling methods into a new and better-performing scheduling policy. The implemented reinforcement learning scheduler only had access to a few, quite basic scheduling methods which resulted in a scheduler with similar behaviour to current adaptive traffic light systems. An interesting continuation of the study would, therefore, be to investigate if improved performance could be reached with the addition of more complex scheduling methods.

In order to reduce the amount of states in the environment, allowing the use of tables to store Q-values, the agent only had access to the number of cars currently waiting in the different lanes in the intersection. For the agent to make more qualified decisions it would need access to more information. For instance, how traffic has behaved in the last hour as well as how long cars have waited in each lane. However, this could lead to a high number of states thus requiring the use of Deep Q-learning.

Lastly, the simulation can be improved by utilising a physical simulation where the cars have physical properties such as speed and length. Achieving a reduced waiting time through means of a physical simulation would further strengthen the case that the reinforcement learning approach used in the study is a viable option for scheduling traffic.

Chapter 6

Conclusion

The thesis sought out to investigate if a traffic scheduler utilising reinforcement learning could perform better than traditional traffic scheduling policies used today, more specifically fixed interval scheduling. A method allowing an agent to choose between different predefined scheduling methods with different characteristics was implemented. The reinforcement agent regularly managed to combine the best characteristics of the different scheduling methods it was given and consequently performed better on average in terms of waiting time compared to the fixed interval scheduler. However, the fact that the agent did not always use the optimal method for each scenario might indicate that the reward function needs slight improvement. Additionally, it had a limited view of the current traffic environment. Further improving the input to the agent as well as providing it with more complex schedulers could potentially lead to even better results.

Bibliography

[1] Monireh Abdoos, Nasser Mozayani, and Ana L. C. Bazzan. "Traffic light control in non-stationary environments based on multi agent Q-learning". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (2011).

[2] Graham Cookson. "INRIX Global Traffic Scorecard". In: (2018).

[3] Weiguo Han, Jinfeng Wang, and Shih-Lung Shaw. "Visual Exploratory Data Analysis of Traffic Volume". In: (Nov. 2006).

[4] Hado V Hasselt. "Double Q-learning". In: (2010).

[5] Anton Jansson and Kristoffer Uggla Lingvall. "Elevator Control Using Reinforcement Learning to Select Strategy". In: (2015).

[6] Guillaume Leduc. "Road Traffic Data: Collection Methods and Applications". In: (Jan. 2008).

[7] Xiaoyuan Liang et al. "Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks". In: (2018). URL: http://arxiv.org/abs/1803.11115.

[8] Azhar Al-Mudhaffar. "Impacts of traffic signal control strategies". In: (2006).

[9] Richard S Sutton and Andrew G Barto. "Reinforcement learning: An introduction". In: (1998).

[10] Richard S Sutton and Andrew G Barto. "Reinforcement learning: An introduction". In: (2011).

[11] Andreas Warberg. "Intelligent Traffic Light Management: Arterial Simulation & Optimization". MA thesis. Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.

[12] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. In: *Machine Learning* (May 1992). URL: https://doi.org/10.1007/BF00992698.

[13] MA Wiering. “Multi-agent reinforcement learning for traffic light control”. In: (2000).