Master's Thesis : Partially Detected Intelligent Traffic Signal Control using Connectionist Reinforcement Learning

Auteur : Geortay, Cyril
Promoteur(s) : Louppe, Gilles
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2019-2020
URI/URL : http://hdl.handle.net/2268.2/9068

University of Liège - Faculty of Applied Sciences

Partially Detected Intelligent Traffic Signal Control using Connectionist Reinforcement Learning

Author
Cyril Geortay

Advisor
Prof. Gilles Louppe

A dissertation submitted in partial fulfillment of the requirements for the degree of MSc in Computer Science and Engineering

Academic year 2019-2020

Abstract

Nowadays, Artificial Intelligence (AI), through Intelligent Transport Systems (ITS), is seen as a major solution to traffic management problems. Among these issues is the optimization of traffic signal control to reduce traffic congestion. The maturity of connected devices is an opportunity to design cheap and efficient traffic light control systems through Vehicle to Infrastructure (V2I) communications.

While traditional ITS use fixed sensors such as cameras or loop detectors, able to detect every vehicle, a V2I implementation faces a real issue: partial detection. Indeed, it cannot be assumed that every vehicle is equipped with the V2I communication technology required to be observable to the ITS.

The present work reports a V2I Intelligent Transport System meant to reduce congestion at traffic light intersections. Another main goal is to decrease the waiting time of each and every road-user at these intersections. This is why the integration into the algorithm of pedestrians and of a priority system for public transports is also analyzed in this work. The system is implemented with a Reinforcement Learning (RL) algorithm, particularly suitable for this kind of problem since it can use varied inputs and does not have to model the underlying dynamics of the environment as it only relies on experience about the efficiency of its actions.

The analyses under different car flows, detection rates and topologies show that the presented algorithm is able to efficiently reduce commute time of road-users compared to currently deployed fixed time systems, even at low detection rates. It is also robust enough to be implemented on real traffic light intersections. Analyzes on road networks with several intersections have shown that the algorithm is able to provide good performances, even without an explicit multi-agent strategy. The integration of a priority system for public transports was successful as well, although an effective integration of pedestrians would require a more complex solution than the one proposed in this work.

Acknowledgements

First, I would like to thank Prof. Gilles Louppe who allowed me to choose one of my own subjects, who followed up my progress regularly and made sure I was on the right path, and who put the tools I needed in my hands.

I am grateful to Matthia Sabatelli, who checked on my work along with Prof. Louppe, for his advice and expertise.

I would also like to thank Joeri Hermans who took the time to solve the issues I had on the GPU cluster.

Many thanks to Philippe, Florence, Chloé, Sven and my father François, who helped to proofread this dissertation.

Finally, thank you to my family and friends, especially to my mother Anne-Catherine, who supported me all along as well as in the last steps of my studies.

Contents

Chapter 1

Introduction

In today's world, with more and more vehicles on the roads, traffic congestion is more than ever a real problem. Part of this congestion is due to a non-optimal management of traffic signals.

Over the past decades, many studies succeeded in reducing queue lengths and waiting time at traffic light intersections. Different methods were developed and deployed in a few cities [1, 2, 3, 4], some of them optimizing fixed-time phases from traffic flow data [1], others implementing dynamic control through traffic sensors [2, 3]. With the growth of Artificial Intelligence (AI) and new technologies during the last few years, the field of possibilities has been significantly increased and more and more Intelligent Transportation Systems (ITS) are being designed as a solution to the traffic light congestion problem. Since the beginning of the century, Reinforcement Learning (RL) has proven to be very efficient in reducing congestion at intersections.

The present work is meant to solve today's issues without overestimating the available financial and technological capabilities. It is based on methods presented in [5] and aims to develop a Connectionist[1] Reinforcement Learning solution for the congestion problem at traffic light intersections. The resulting ITS is analyzed under different configurations and on a larger scale. Its robustness is tested against several factors and a deployment case.

1.1 Problem statement

The Internet of Things (IoT) has already enabled a huge amount of improvements in our everyday lives. Traffic light control can also capitalize on this for alleviating traffic congestion with technologies that allow for cheap deployment and maintenance. Indeed, most of the Adaptive Traffic Control Systems (ATCS) proposed and developed in recent years are still relying on vehicle state and quantity information that is either very difficult to obtain or that requires an expensive equipment such as cameras or loop detectors. Hence, in most cities, traffic lights are still controlled by fixed-time-phases

[1]Connectionism is a field of cognitive science that aims to explain mental phenomena with the help of Artificial Neural Networks (ANN). Connectionist Learning, which predates Deep Learning by 10 to 20 years, uses interconnected networks of simple and often uniform units.

policies. With the generalisation of smartphones and other connected devices all over the world, other traffic light control methods could take advantage of these new technologies to replace traffic sensors.

The IoT approach consists in placing the technology in the driver's seat instead of setting it up at each traffic light intersection. It could either be external to the vehicle (e.g. a free app developed by a city or state) or integrated in it (e.g. DSRC technology [6]). Moreover, unlike fixed cameras or loop detectors, this kind of connected systems allows for tracking vehicles (position, speed,...), thus providing a greater amount of data that can be used in the state representation of the intersection.

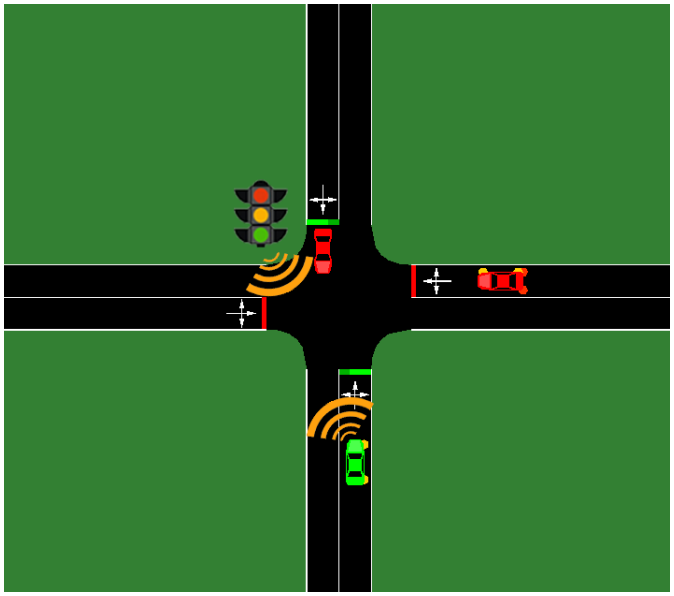


Figure 1.1: Illustration of an Intelligent Transport System with partial detection.

This change in the mean of communication leads to another issue: it cannot be expected that every user is able or agrees to get the proposed technology. It follows that the implemented ITS solution will have to take decisions on traffic light phases in a partially observable environment, as some vehicles will not be detected (detection/penetration rate lower than 100%). Figure 1.1 pictures a single traffic light intersection controlled by an ITS with partial detection. The RL agent is represented by the traffic light symbol, detected and undetected vehicles appear respectively in green and red.

In this work, the two main and strongly linked objectives are the following:

- To design a single-agent RL algorithm that is able to reduce traffic congestion, even with a low detection rate, and thus to provide a cost-effective implementation that can replace fixed time traffic signal control.
- To reduce the average waiting time of all road-users. This calls for the integration of a priority system for public transports and the consideration of pedestrians in some intersections.

The outlined solution could be considered as permanent or as a transition to a more efficient approach that would require a detection rate close or equal to 100% (e.g. Virtual Traffic Lights [7]). In order to enhance the performances of the algorithm and possibly to speed up the transition process to another Intelligent Transport System, it is discussed how road-users can be encouraged to switch to the communication technology required by the solution to increase detection rate.

This work also analyzes how sensitive the proposed ITS is to several factors such as traffic flow and detection rate and how well it can scale up to a network of intersections. A deployment case is also evaluated thanks to a scenario modelling a real traffic light intersection in Luxembourg and using historical traffic data. Finally, a study of the integration of pedestrians and of a priority system for public transports is conducted.

The algorithm proposed in this thesis relies on Connectionist Reinforcement Learning, particularly Q-Learning, as it has proven to provide very good performances in the past. Indeed, the problem that is tackled is a control problem that can be modelled as a Markov Decision Process (MDP). It is therefore well suited for this kind of algorithms.

Chapter 2

Related work

Traffic signal control using Reinforcement Learning has been studied for more than 20 years. In 1994, Mikami et al. proposed a Genetic RL solution to the traffic light congestion problem over a road network by trading between two conflicting objectives for each agent: optimize the local control plans while simultaneously cooperating with the other agents, using respectively Reinforcement Learning and combinatorial optimization [8].

In 2000, Wiering used multi-agent model-based Q-Learning, considering traffic lights and vehicles as agents sharing the same value function (Co-Learning), to optimize traffic light settings and re-route cars [9]. Wiering et al. then proposed an improvement of this implementation in the form of a bucket algorithm (communication between agents), and tested the solution on their Green Light District simulator [10]. Bingham developed a neural network with a RL algorithm to fine-tune the parameters of a traffic signal controller that uses fuzzy logic [11].

With the rediscovery and generalization of neural networks, Connectionist Reinforcement Learning algorithms proved to be valuable schemes. In 2010, Arel et al. designed a centralized multi-agent RL algorithm with a feedforward neural network for value function approximation [12], based on the Longest Queue First Maximal Weight Matching (LQF-MWM) algorithm proposed by Wunderlich et al. [13]. Another multi-agent algorithm was proposed in 2019 by Wang et al., who used Cooperative double Q-Learning (Co-DQL) and the Upper Confidence Bound (UCB) policy to avoid overestimation along with a fully connected neural network [14].

The recent improvement of Deep Learning and GPU has given Deep Reinforcement Learning an important role to play in many fields, including intelligent traffic signal control. In 2016, Van der Pol et al. introduced a scalable Deep Q-Learning approach using transfer planning, max-plus coordination algorithm and convolutional layers in Q-Network [15]. This paper was a source of inspiration for Gao et al., who proposed a single-agent DRL solution using real-time machine-crafted features from raw traffic data, instead of often used human-crafted features which ignore some traffic information [16]. In 2019, Chu et al. developed a decentralized multi-agent DRL algorithm using advantage actor critic (A2C) agents instead of widely used Q-Learning agents

and Long-Short Term Memory (LSTM) layers in their neural network [17].

The present work, although considered as a baseline, is strongly inspired from the techniques exposed in 2018 by Zhang et al. [5]. They exploited the development of Dedicated Short Range Communications (DSRC) technology [6], that is going to be implemented in US cars in the near future, as a mean of Vehicle to Infrastructure (V2I) communications. In this paper, a Q-Learning algorithm is proposed as a cost-effective transition solution to Virtual Traffic Lights (VTL): an infrastructure-free traffic control scheme that leverages the presence of DSRC-based Vehicle to Vehicle (V2V) communications, eliminating the need of any traffic light systems [7]. In October 2019, further study of Partially Detected Intelligent Traffic Signal Control (PD-ITSC) was led by Zhang et al. about the environmental adaptation of different Reinforcement Learning algorithms [18].

Not only does this work re-assess the approach from [5], it also adds some more elements of analysis on the methods described, focusing only on a Q-Learning algorithm, and evaluates two relatively new aspects of traffic signal control: the integration of pedestrians and of a priority system for public transports.

Chapter 3

Reinforcement Learning

Reinforcement Learning (RL) is a field of machine learning that consists of an autonomous agent evolving in an environment and perceiving information about its current state. From these observations, it can decide to perform some actions following a policy. Using this policy, the goal of the agent is to maximize an expected cumulative reward over the course of interaction with the environment. For this purpose, the agent iteratively collects a state observation, performs an action based on it in order to maximize its expected cumulative future reward, and gets the reward associated to this action in this state. The process is illustrated in Figure 3.1.

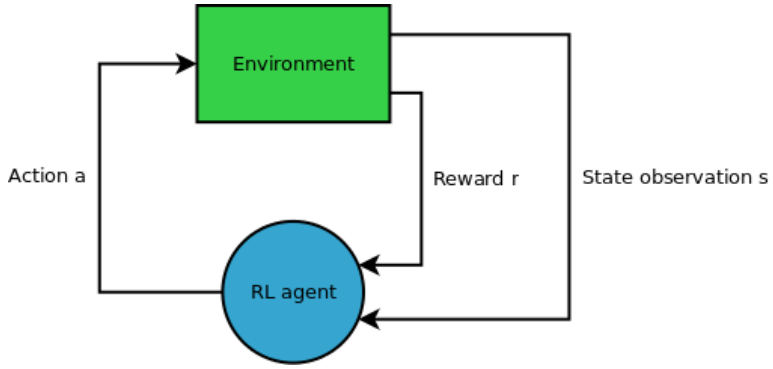


Figure 3.1: Reinforcement Learning agent interacting with its environment.

Based on the collected rewards, the RL agent optimizes its policy, either directly (policy-based RL) or by optimizing a value function that estimates the expected future reward that is obtained by performing a certain action in a certain state (value-based RL). In the case of a value-based algorithm, the action chosen by the agent is the one that maximizes the value function. The value-based approach that is used in this thesis is called Q-Learning.

3.1 Neural Networks

For more details about Neural Networks, the current section is mainly based on the lectures of Louppe [19].

In Deep and Connectionist value-based Reinforcement Learning, the value function is approximated by a Neural Network (NN). One should introduce the architecture and mechanisms of the main NN used in the connectionist approach, i.e. the fully connected feedforward network or Multi-Layer Perceptron (MLP).

3.1.1 The fully connected feedforward network

The fully connected feedforward network or Multi-Layer Perceptron (MLP) is the simplest form of Neural Network. Still, this network is able to approximate any continuous function.

The building block of any MLP is called the neuron or perceptron. This neuron takes some inputs x_i, whose contributions are weighted by some weights w_i, and outputs a value added to a potential bias term b. This output value then goes through an activation function $a(x)$, resulting in:

$$h = a\left(\sum_i w_i x_i + b\right) \ , \tag{3.1}$$

or, in terms of tensors:

$$h = a(\mathbf{w}^T \mathbf{x} + b) \ . \tag{3.2}$$

The most common activation function in modern Neural Networks is the Rectified Linear Units (ReLU) function:

$$\mathrm{ReLU}(x) = \max(0, x) \ . \tag{3.3}$$

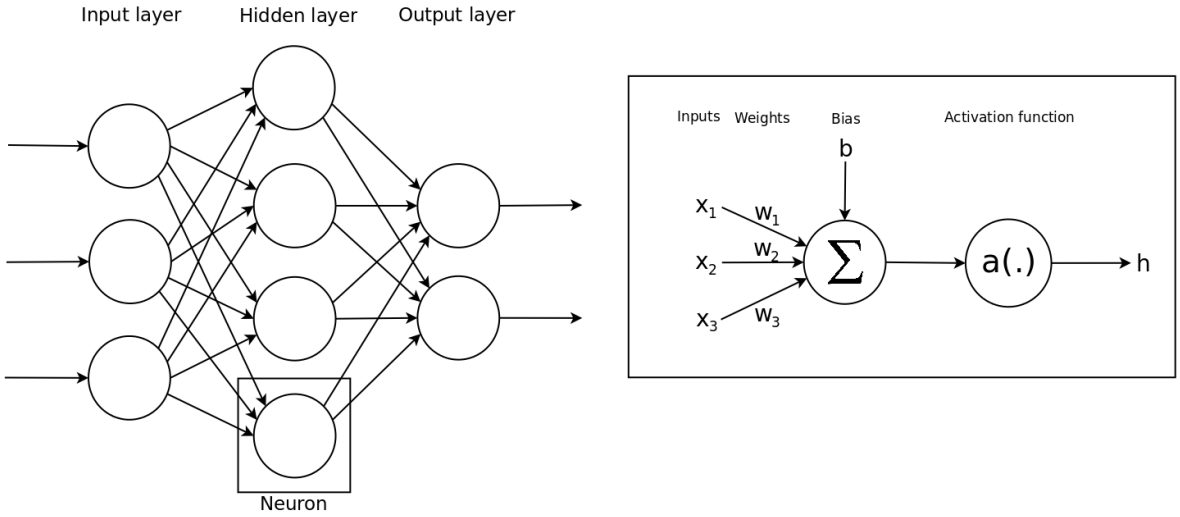


Figure 3.2: A fully connected feedforward neural network, or MLP, with three input features, four hidden neurons and two outputs, along with an illustration of the calculation performed in a neuron.

Neurons can be composed in parallel to form layers (called fully connected layers). These layers can then be composed in series to form the MLP. Every MLP consists in

one input layer, one or several hidden layers and one output layer. Each neuron in a layer connects to all the neurons in the next layer. This explains the adjective "fully connected". An example of MLP is represented in Figure 3.2. The arrows are pointing to the direction of the forward pass.

3.1.2 Gradient descent and backpropagation

A Neural Network is used to approximate a model that represents a relation in some data. In order to measure the correctness of the network, its predictions are compared to the expected outputs over a data set through a loss function $\mathcal{L}(\theta)$, where θ are the parameters of the network (the weights and biases over the whole network).

The goal is to minimize the loss function. Gradient descent uses local linear information to iteratively move towards a local minimum. For initial parameters θ_0, a first-order approximation can be defined as:

$$\hat{\mathcal{L}}(\theta_0 + \epsilon) = \mathcal{L}(\theta_0) + \epsilon^T \nabla_\theta \mathcal{L}(\theta_0) + \frac{1}{2\alpha}\|\epsilon\|^2 \ . \tag{3.4}$$

$\hat{\mathcal{L}}(\theta_0 + \epsilon)$ is minimized if:

$$\nabla_\epsilon \hat{\mathcal{L}}(\theta_0 + \epsilon) = 0 = \nabla_\theta \mathcal{L}(\theta_0) + \frac{1}{\alpha}\epsilon \ , \tag{3.5}$$

which results in the best improvement for the step $\epsilon = -\alpha \nabla_\theta \mathcal{L}(\theta_0)$.

Therefore, parameters of the networks are updated iteratively following the update rule:

$$\theta_{t+1} \coloneqq \theta_t - \alpha \nabla_\theta \mathcal{L}(\theta_t) \ , \tag{3.6}$$

where α is the learning rate.

Minimizing the loss function $\mathcal{L}(\theta)$ with (3.6) requires the gradient $\nabla_\theta \mathcal{L}(\theta_t)$. This is calculated thanks to the derivatives $\frac{d\mathcal{L}}{d\mathbf{W}_k}$ and $\frac{d\mathcal{L}}{d\mathbf{b}_k}$, where $\mathbf{W}_k$ and $\mathbf{b}_k$ are the parameters of one of the L layers of the Neural Network, with $k = 1, ..., L$.

The chain rule states that:

$$(f \circ g)' = (f' \circ g)g' \ . \tag{3.7}$$

Since a Neural Network is a composition of differentiable functions, the derivatives of the loss function can be evaluated through a backward pass by applying the chain rule recursively. This is called backpropagation.

As an example, Figure 3.3 shows the computational graph of a simple two layer network where:

- White nodes correspond to input, output and intermediate values.
- Green nodes correspond to model parameters (in this case only weights).

- Blue nodes correspond to intermediate operations.

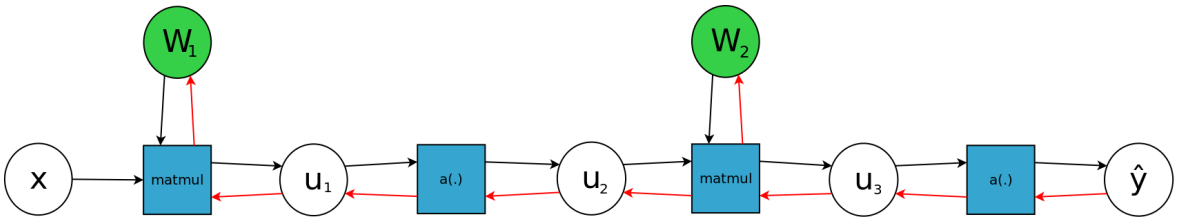


Figure 3.3: Computational graph of a simple two layer network, along with an illustration of the backward pass.

Following the backward pass (represented by red arrows on Figure 3.3), $\frac{d\hat{y}}{d\mathbf{W}_1}$ is obtained by:

$$\frac{d\hat{y}}{d\mathbf{W}_1} = \frac{\partial \hat{y}}{\partial u_3}\frac{\partial u_3}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial \mathbf{W}_1} = \frac{\partial a(u_3)}{\partial u_3}\frac{\partial \mathbf{W}_2^T u_2}{\partial u_2}\frac{\partial a(u_1)}{\partial u_1}\frac{\partial \mathbf{W}_1^T u_1}{\partial \mathbf{W}_1} \ . \tag{3.8}$$

3.2 Markov Decision Processes

For more details about Markov Decision Processes, the current section is based on the lectures of Louppe [20].

Reinforcement Learning tackles problems that can be modelled as a Markov Decision Process (MDP). It enables to define formally the environment in which the RL agent is evolving and the interactions between the two. A Markov Decision Process is modelled as a 4-tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.
- $P(s'|s,a)$ is the transition function and denotes the probability to reach state s' by performing action a in state s.
- $R(s)$ is the reward function and denotes the immediate finite reward value obtained for reaching state s.

A particularity of MDP is that future and past states are independent:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = P(s_{t+1}|s_t, a_t) \ . \tag{3.9}$$

A policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ maps states to actions. The discounted utility of a sequence of rewards is given by:

$$V([r_0, r_1, r_2, ...]) = r_0 + \gamma r_1 + \gamma^2 r_2 + ... \ , \tag{3.10}$$

where $0 < \gamma < 1$ is the discount factor and reflects the fact that sooner rewards probably have greater utility than later rewards.

From (3.10), it comes that the expected discounted utility of a sequence of actions generated by a policy π, starting from a given state s, is defined by:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]\Bigg|_{s_0=s} , \tag{3.11}$$

where the expectation is taken with respect to the probability distribution over the sequence of states generated by π and s, through the transition function.

This aims at finding an optimal sequence of actions, i.e. an optimal policy π^* that maximizes the expected discounted utility, such that:

$$\pi^*(s) = \arg\max_{\pi} V^{\pi}(s) \ . \tag{3.12}$$

Therefore, assuming an optimal policy π^*, the expected discounted utility of a state corresponds to the immediate reward for that state plus the expected discounted utility of the next state. This next state is obtained by choosing the optimal action. This relation is described by the Bellman equation:

$$V^{\pi^*}(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s,a) V^{\pi^*}(s') \ . \tag{3.13}$$

This relation exploits Equality (3.9).

3.3 The Q-Learning algorithm

For more details about Q-Learning, the current section is based on the work of Watkins et al. [21].

In the Q-Learning algorithm, the agent learns a value function called the Q-value function $Q(s_t, a_t)$, which returns the maximum expected cumulative discounted future reward (i.e. the maximum expected discounted utility), given a state observation s_t and an action a_t, at time t. Each time step t is characterized by the set of values $[s_t, a_t, r_t, s_{t+1}]$, which corresponds to performing an action a_t in state s_t and receiving a reward r_t for reaching state s_{t+1}. Relying on the Bellman equation (3.13), the maximum expected cumulative discounted future reward is defined by:

$$Q(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\right] \ . \tag{3.14}$$

Since time horizon for the traffic light control problem is infinite, the agent trains on a chosen number of episodes of a chosen number of time steps.

In traditional Q-Learning, the agent directly updates its Q-value at each time step following the update rule:

$$\begin{aligned} Q(s_t, a_t) &\coloneqq (1-\alpha) Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})) \\ &= Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \ , \end{aligned} \tag{3.15}$$

where α represents the learning rate.

In Connectionist or Deep Q-Learning, as the state space can be too complex for a discrete representation, a neural network (Q-Network) is used to approximate the Q-function. It is thus the Q-function $Q_\theta(s_t, a_t)$ that is updated at each time step using backpropagation with a loss calculated with the following target value:

$$r_t + \gamma \max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1}) \ , \tag{3.16}$$

where θ denotes the parameters of the Q-Network, which are the ones that are updated.

In this work, in order to stabilize the learning process, three features have been added to the basic algorithm:

- Double Q-Learning (DQL), that consists in using two distinct networks Q_θ and $Q_{\theta'}$ to approximate the Q-function. Q_θ corresponds to the regular Q-Network, backpropagated every time step, while $Q_{\theta'}$ corresponds to the target network, used to compute the target values and updated with the parameters θ of Q_θ (i.e. $\theta' \leftarrow \theta$) periodically after a determined number of time steps. Without a target network, target values would always be changing along with the parameters of the regular Q-Network, which makes convergence more difficult. With this approach, instead of using a target value as defined in (3.16), the latter is computed as:

  $$r_t + \gamma \max_{a_{t+1}} Q_{\theta'}(s_{t+1}, \arg\max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1})) \ . \tag{3.17}$$

  This new definition enables to prevent overestimation of the Q-values. Indeed, choosing an action and estimating its corresponding Q-value with different networks allows to decorrelate their noise.

- The use of a replay buffer, storing a certain number of transitions $[s_t, a_t, r_t, s_{t+1}]$, from which a batch is randomly drawn at each time step for backpropagation. This allows to break time correlation between transitions, which is a recurrent problem of on-line Q-Learning.

- Scheduling on the learning rate used for backpropagation. This consists in choosing a high value for the learning rate at the beginning of training in order to quickly converge to a good reward, and then decreasing it to prevent oscillations.

Chapter 4

Methodology

4.1 General design

In a classic traffic signal intersection, traffic lights are located at each approach and coordinate into phases. Each phase is composed of a combination of light colors (green, red or amber) defined to ensure all road-users' safety. Figure 4.1 shows an example of the different traffic light phases of a simple intersection.

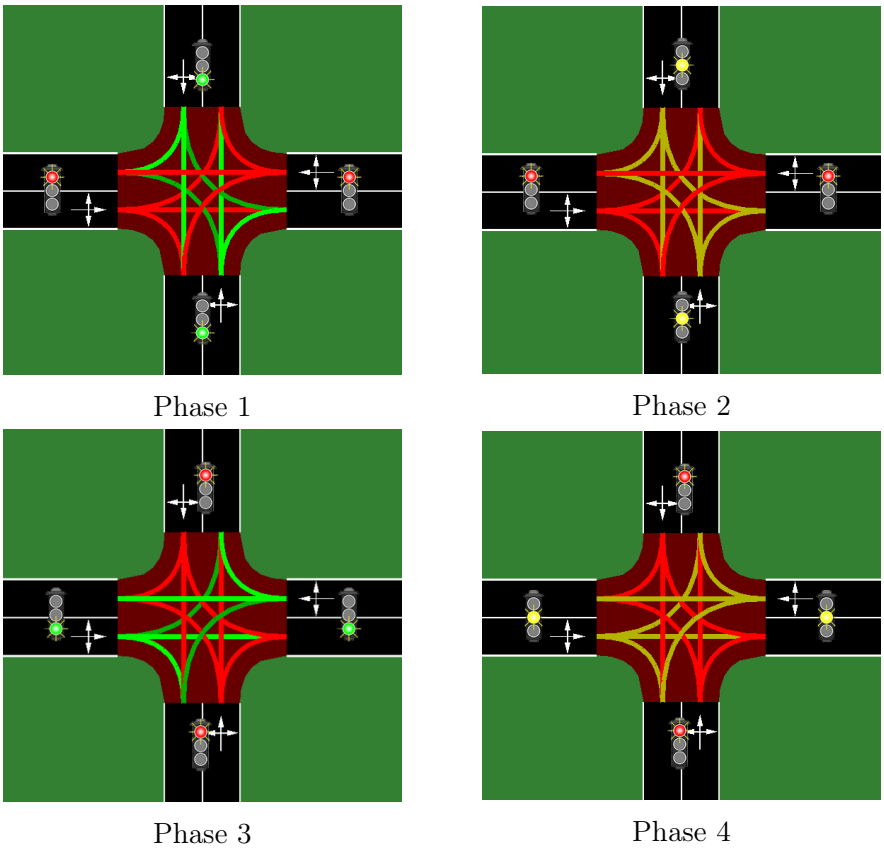


Figure 4.1: The 4 traffic light phases of a simple intersection.

The ITS proposed in this work is illustrated in Figure 4.2. At each time step, it loads the current state representation of the intersection, processes it with its Reinforcement

Learning algorithm, and outputs an action to take: either stay at the current phase (the output Q-value of the Q-Network is maximized for the action of staying at the current phase: $Q_{stay} > Q_{switch}$), or switch to the next one ($Q_{stay} < Q_{switch}$). For the sake of safety and realism, a traffic light phase has a minimum and maximum duration time before switching to the next phase. This means that before reaching minimum phase time, the agent cannot switch to the next phase whatever the maximum Q-value, and once the maximum phase time is reached, the agent is forced to switch to the next phase.

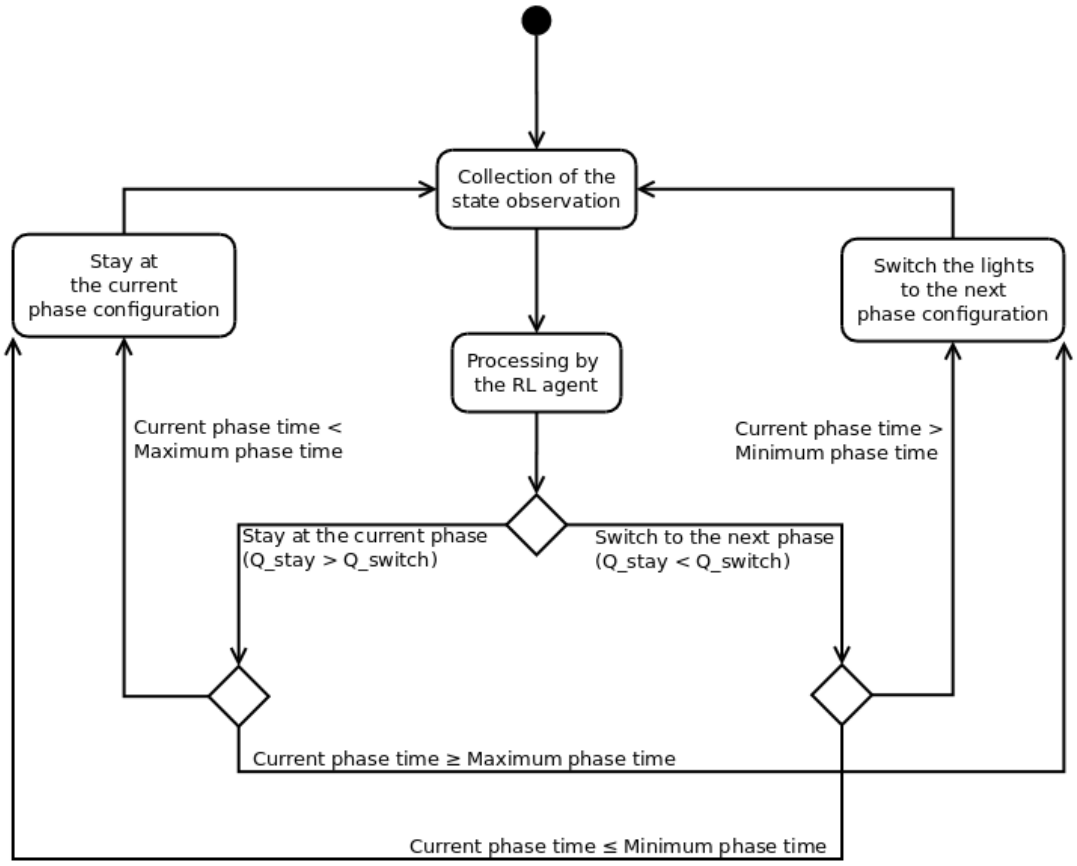


Figure 4.2: General design of the Intelligent Transport System.

4.2 Parameters of the Q-Learning algorithm

4.2.1 Q-Network

The Reinforcement Learning agent used in the ITS is designed with a Q-Learning algorithm, using a Q-Network as Q-function. This Q-Network has fully connected layers with Rectified Linear Unit (ReLU) activation. It takes a state observation in entry and outputs a Q-value for each action. The policy is such that the action corresponding to the maximum Q-value is chosen by the agent. Detailed information about the architecture, hyperparameters and training of the Q-Network is provided in appendix A.

4.2.2 State representation

The state observation that is given as input to the Q-Network is a vector of features. The information it contains is listed in Table 4.1.

Feature	Description
Number of detected vehicles	Number of detected vehicles for each lane/approach.
Distance to the nearest detected vehicle	Normalized distance to the nearest detected vehicle for each lane/approach. If there are no vehicles in a lane, set to the normalized lane length (i.e. 1).
Current phase time	Normalized number of seconds elapsed since the beginning of the current phase.
Amber phase	1 if we are in a phase where at least one of the lights is amber, 0 otherwise.
Current day time	Normalized number of seconds elapsed since the beginning of the day.

Table 4.1: Detailed description of the state representation.

Notice that a lane or approach is not to be confused with the entire incoming road segment. A road segment is demarcated by sidewalks or aisles while there can be several lanes (going in the same direction or not) in a road segment.

In order to differentiate phases, the signs of the "Number of detected vehicles" and of the "Distance to the nearest detected vehicle" features are changed if the light that is shown to the corresponding lane is red or amber (meaning negative values for red or amber colors and positive values for green color). Along with the use of ReLU activation functions, this sign change also enables to activate different parts of the neural network.

In order to speed up learning, most of the input features are normalized, which means, in this work, that their possible range of values is reduced to an interval between 0 and 1 (e.g. the "Current day time" feature is divided by 86400, i.e. the number of seconds in a day).

4.2.3 Actions

The actions that can be taken by the RL agent at each time step are either:

- To keep the current phase configuration active.
- To switch to the next phase configuration.

Hence, the Q-Network has two output nodes corresponding to the Q-values of these two actions.

4.2.4 Reward

In order for the agent to learn the right parameters, the reward must be chosen as a measure inversely translating the amount of congestion at the intersection. This can be written as a function of the delay $t_{ITS} - t_{min}$ of a vehicle passing through the intersection, where t_{ITS} and t_{min} represent respectively the travel time under a certain ITS and the physically possible minimum travel time of a vehicle through the intersection.

Starting from this, the distance d through the intersection can be expressed as:

$$d = \int_0^{t_{ITS}} v_{ITS}(t)dt = v_{max}t_{min} \ , \tag{4.1}$$

where $v_{ITS}(t)$ is the speed of the vehicle passing through the intersection at time t, and v_{max} is the maximum reasonable speed corresponding to the minimum between the maximum speed of the vehicle involved v_{max}^{veh} and the speed limit of the road v_{lim} (so, in most cases, v_{max} is the speed limit):

$$v_{max} = \min(v_{max}^{veh}, v_{lim}) \ . \tag{4.2}$$

The delay can thus be defined by:

$$\begin{aligned} t_{ITS} - t_{min} &= \int_0^{t_{ITS}} 1dt - \frac{1}{v_{max}} \int_0^{t_{ITS}} v_{ITS}(t)dt \\ &= \frac{1}{v_{max}} \int_0^{t_{ITS}} v_{max} - v_{ITS}(t)dt \ , \end{aligned} \tag{4.3}$$

where, theoretically, $v_{max} - v_{ITS}(t) \geq 0, \forall t$.

Therefore, at each time step, the goal is to minimize:

$$\frac{1}{v_{max}}[v_{max} - v_{ITS}(t)] \ . \tag{4.4}$$

This expresses a penalty. As the reward is meant to be maximized, it can rather be defined by the opposite relation:

$$r_t = \frac{v_{ITS}(t) - v_{max}}{v_{max}} \ . \tag{4.5}$$

This reward corresponds to a single vehicle. The total reward, calculated for the entire intersection at a certain time step t, is equivalent to the average of the rewards corresponding to the vehicles currently passing through the intersection. It is also interesting to notice that this reward could not have been computed with traditional traffic sensors, such as cameras or loop detectors, as it requires ability to track vehicles.

However, the reward as defined above can only be calculated from a simulated environment, as it bases on information collected from undetected vehicles. This means that, once deployed, the agent will not be able to train anymore, and thus to adapt further to its assigned intersection. This problem can be solved by using a partial reward only considering detected vehicles.

4.3 Simulation of the environment

The training and testing environments in this work are simulated with the Simulation of Urban MObility (SUMO) simulator [22]. SUMO is an open source, microscopic, space-continuous, and time-discrete traffic flow simulation platform that includes the simulation application itself as well as supporting tools. It has been widely used during the last decade in various traffic management and vehicular communication studies, including adaptive traffic signal control.

The architectures of the environments created for this thesis were designed with the *netedit* tool, a GUI for road network edition. Communication between the simulator and the RL agent was enabled through the *Traffic Control Interface* (TraCI) API.

As SUMO is a time-discrete simulator, each time step in the simulator corresponds to one second in reality. This is a realistic frequency at which the implemented ITS could collect state observations and take actions once deployed on a real intersection.

Chapter 5

Performance analysis on a simple intersection

A first step in the analysis of the performances of the Q-Learning algorithm implemented in this thesis is to test it in different situations within a simple environment. This chapter focuses on two points: the general performance results of the algorithm on a simple intersection, and its sensitivity and influenceability to several factors.

5.1 Description of the environment

The simple intersection on which the algorithm is tested is meant to be as general as possible. Hence, as shown in Figure 5.1, the chosen intersection is a cross-like intersection with incident roads composed of a single incident 100 meters long lane. A vehicle coming from any lane can either turn left, right or keep going straight.

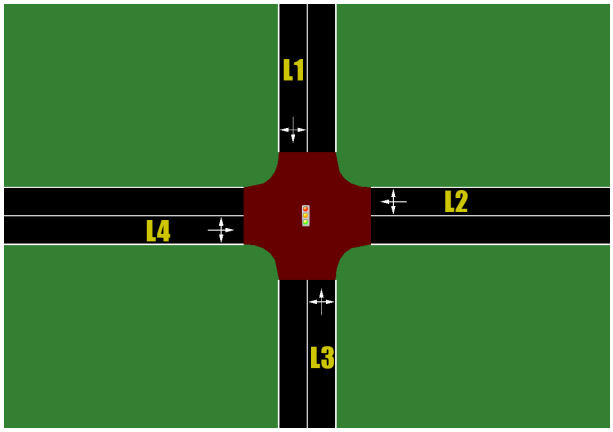


Figure 5.1: General simple intersection on which the algorithm is tested (L=lane).

The light phases are as shown in Figure 4.1 and explained in Table 5.1. The results obtained with a minimum phase time lower than 10 seconds for phases 1 and 3 could lead to better performances, as the algorithm would benefit from a better reaction time and light maneuverability. However, for the sake of realism, this value has been maintained at 10 seconds.

	L1	L2	L3	L4	Duration
Phase 1	G	R	G	R	Minimum: 10″ Maximum: 50″
Phase 2	A	R	A	R	3″
Phase 3	R	G	R	G	Minimum: 10″ Maximum: 50″
Phase 4	R	A	R	A	3″

Table 5.1: Description of the 4 phases of the simple intersection (L=lane, colors correspond to light colors: G=green, R=red, A=amber).

For a complete analysis, the algorithm is tested against different traffic flow configurations that are presented in Table 5.2. These traffic flows are generated randomly: every time step, the generation or non-generation of a vehicle for each possible path in the intersection is determined by drawing from a Bernoulli distribution[1]. The choice of non-uniform traffic flows is motivated by the fact that the two axes of a typical cross-like intersection rarely have the same traffic flows (e.g. a main arterial road crossed by some smaller, less frequented roads). A video of the different traffic flows (low, medium, high and very high) is available at [23]. The very high traffic flow illustrates an extreme case where congestion is so high that traffic is just a continuous queue.

		Traffic flow per lane (in veh/s)			
		West	**North**	**East**	**South**
Uniform	**Low**	0.01	0.01	0.01	0.01
	Medium	0.05	0.05	0.05	0.05
	High	0.1	0.1	0.1	0.1
	Very high	0.2	0.2	0.2	0.2
Non-uniform	**Medium - Low**	0.05	0.01	0.05	0.01
	High - Low	0.1	0.01	0.1	0.01
	High - Medium	0.1	0.05	0.1	0.05

Table 5.2: Traffic flow configurations of the simple intersection used in the tests.

The algorithm is also tested against 4 detection rates: 100%, 70%, 50% and 20%. The detection or non-detection of a vehicle is determined by drawing from a Bernoulli distribution.

[1]The Bernoulli distribution is the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability $q = 1 - p$.

5.2 Performances

5.2.1 Convergence of the algorithm

Before looking at the actual performances of the Q-Learning algorithm, it is interesting to examine how it converges to an optimized reward during the training process. Indeed, when analyzing the training curve of the algorithm at uniform medium traffic flow (the training curves for all the traffic flow configurations can be seen in appendix B) in Figure 5.2[2], it can be noticed that the reward stabilizes to a better score than a fixed time system after only 50 episodes. This phenomenon could be caused by a convergence to a solution that is too simple, meaning that the agent acquired little new information.

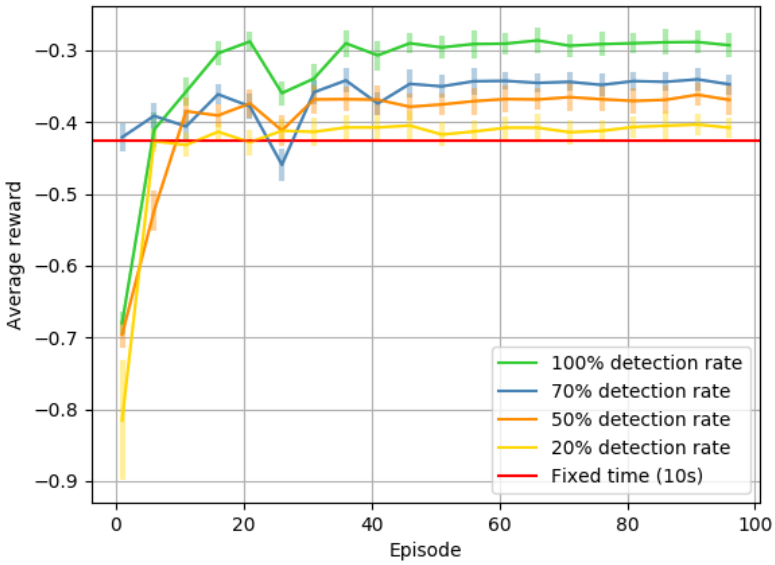


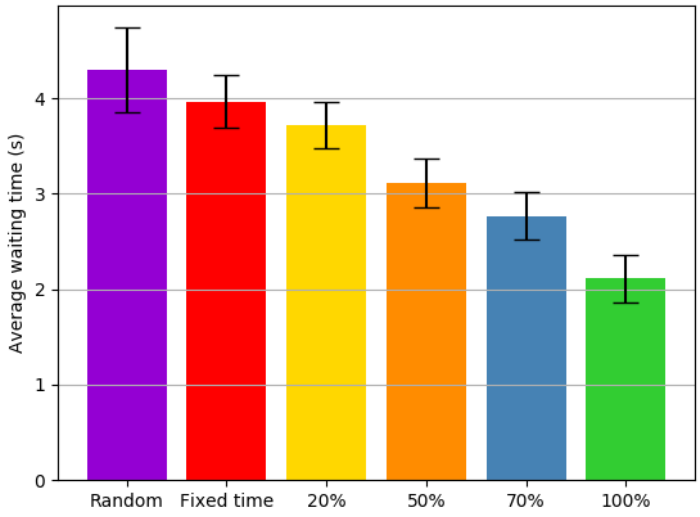
Figure 5.2: Training curve at uniform medium traffic flow, for different detection rates.

In order to exclude this hypothesis, the performances of the algorithm in terms of waiting time have been compared to a random policy, with uniform probability of choosing one action or the other at each time step. The convergence has also been analyzed for simpler algorithms: on-line Q-Learning, and reduction of the amount of exploration by reducing the initial value (from 1 to 0.1) or the number of decay steps (from 100000 to 10000) of ϵ for the ϵ-greedy policy.

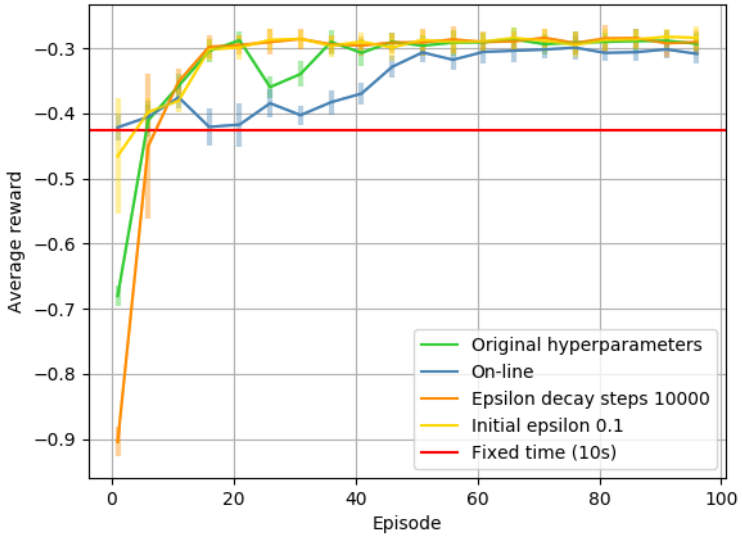
The results of these two experiments are presented in Figure 5.3. Figure 5.3a shows that the algorithm is learning something significant compared to a system making completely random choices and provides performances that are better than a fixed

[2]The curves are starting at two different value zones of the reward: around the fixed time baseline, and around a worse value. This is due to the fact that, when initialized, the Q-Network is not tuned at all and will almost always have its maximum Q-value corresponding to the same action: either "stay at the current phase" or "switch to the next phase". The first one corresponds to phases of 50 seconds and is the worst choice, and the second one corresponds to phases of the minimum phase time: 10 seconds, which is the phase time of the fixed time system used as baseline.

time light controller. On the other hand, Figure 5.3b shows that simpler algorithms also converge to the same reward after 50 episodes. Hence, the problem of managing traffic lights is simple and does not require a lot of training episodes to find an effective solution.



(a) Comparison of the performances of the algorithm, for different detection rates (%), against a random policy.



(b) Comparison between the training curves of the algorithm and other simpler versions of it.

Figure 5.3: Results of the convergence and learning tests.

Note that the error bars in Figures 5.2, 5.3a and 5.3b represent the standard deviation of the average reward or the standard deviation of the average waiting time

following if the figure represents average reward or waiting time. This is the case for all the figures with error bars in this dissertation.

5.2.2 General results

Now that it has been proven that the Q-Learning algorithm is converging to a solution, performance results on a simple intersection can be analyzed with confidence. Figure 5.4 displays the average reward and average waiting time obtained in the traffic flow configurations exposed in section 5.1. The different phase times chosen in each traffic flow configuration for the fixed time system correspond to the optimal ones. These phase times are calculated as a function of the traffic flow in each road segment, in the bounds imposed by minimum and maximum phase times. This means that, especially for more realistic non-uniform traffic flow configurations where the exact flow of each lane is not supposed to be known, the baselines are ideal cases that shall be approached but not reached in reality. A video comparing the performances of fixed time control and of the algorithm with 100% detection rate for all traffic flows is available at [24].

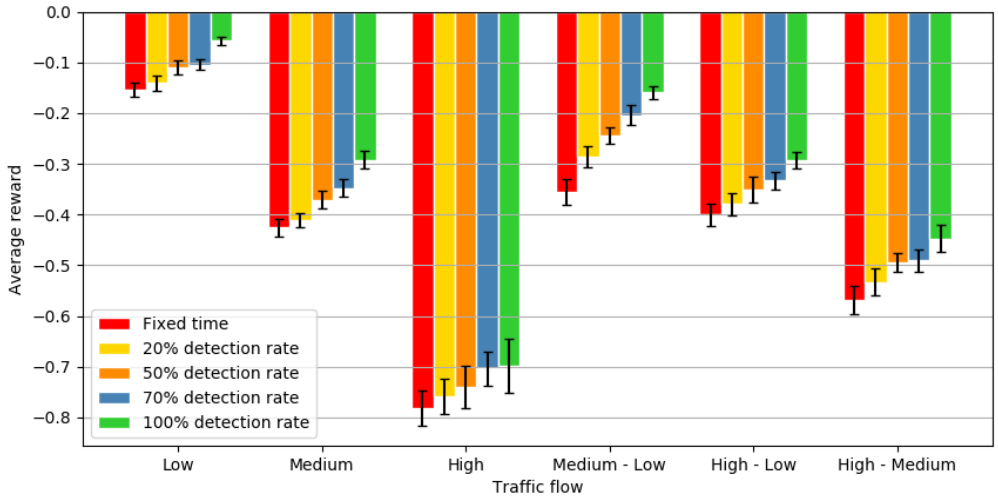
First of all, as the reward is none other than a measure of decongestion (as demonstrated in section 4.2.4), Figures 5.4a and 5.4b show the strong correlation between congestion and waiting time: a low reward corresponds to a high waiting time and vice versa. Hence, analyzing the performances only on the basis of the waiting time is not a loss of information. That is what will be done from now on.

It can also be noticed that the performances are better than the baselines in every configuration, even for low detection rates. However, in situations where the detection rate is 20% (for low, medium, high-low and high-medium configurations), or in a high traffic flow configuration, the error bars of the baseline and the algorithm overlap. This means that it is possible to have better performances with a fixed time system than with the implemented ITS in some situations.

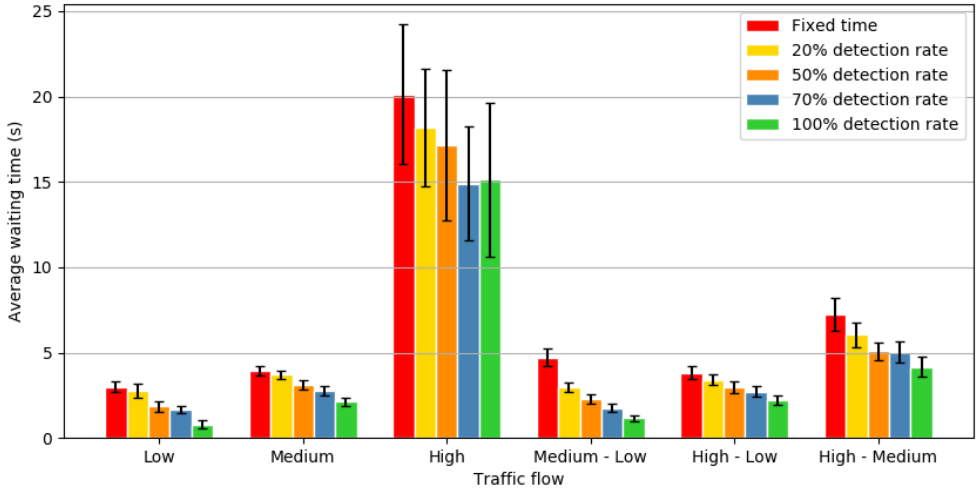
The order of magnitude of the waiting times is pretty similar between a uniform traffic flow configuration and a non-uniform one with the same traffic flow in the less frequented approaches (medium-low is more similar to low than to medium, etc.). The algorithm can thus adapt well to situations where only some approaches of the intersection suffer from high traffic flow, which is often the case in the real world.

The high standard deviation, especially for the average waiting time, in a uniform high traffic flow configuration can be explained by the fact that the problem becomes too complex for the simple state representation that has been chosen[3]. Hence, for a certain state, the algorithm could choose the right action to take, and for a slightly different state that seems similar with respect to the state representation, the algorithm would act wrong.

[3]For example, the algorithm might perform better when including a notion of vehicle position in each approach in the state representation, along with convolutional layers as a first forward stage of the Q-Network, for this kind of input. This scheme is not studied in this work.



(a) Average reward

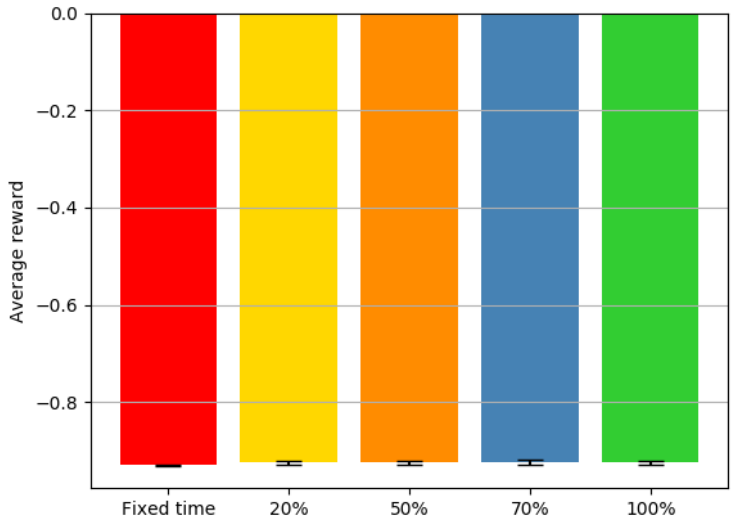


(b) Average waiting time

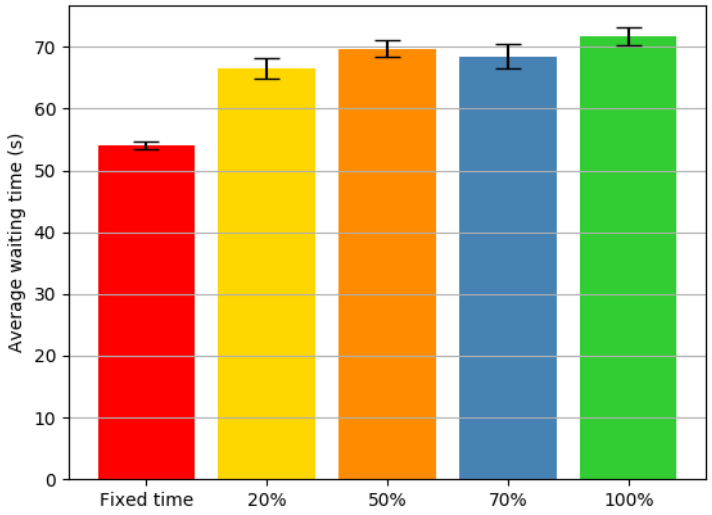
Figure 5.4: General performance results on a simple intersection for different traffic flows and detection rates.

From here, there is one traffic flow configuration that has not yet been studied: uniform very high traffic flow. This is because the results for this configuration are very different and not comparable to the others, as can be seen in Figure 5.5. Indeed, in this situation, the traffic flows are so high that the queues are continuous and there is no intelligent solution to learn, the best one being to alleviate the queues along each axis in a periodic manner, as it is done with fixed time control. This is why the baseline has a lower average waiting time than the algorithm and the standard deviations are so low. There is nothing complex about the problem anymore, a fair amount of vehicles just have to pass through the intersection at any green phase. It is also interesting to notice, when looking at Figure 5.5a, that, unlike for waiting time, the algorithm has learned the same reward than the one obtained with fixed time control, which

corresponds to the best one. The correlation between congestion and waiting time is thus lost when reaching unmanageable traffic flows.



(a) Average reward



(b) Average waiting time

Figure 5.5: Performance results for a uniform very high traffic flow and different detection rates (%).

5.3 Sensitivity

Little are the chances that the environment under which an ITS has been trained is the same as the one on which it is meant to be deployed. This is why a study of

the sensitivity of the algorithm against different parameters is a necessary step in the validation stage.

5.3.1 Normalization

A first question that one could think of when using normalized data in the state representation of an RL algorithm is the effect it has when the reference values for normalization are changing. In the present case, when browsing through Table 4.1, as a day will always be composed of 86400 seconds, the only two values that are concerned are the distance to the nearest detected vehicle and the current phase time. Hence, the question is: "Is normalizing these two values a good idea with respect to adaptability to other lane lengths and maximum phase times?". Figure 5.6 compares the average waiting times obtained for different lane lengths and phase times, at uniform medium traffic flow and 100% detection rate, in the four possible lane length and phase time normalization configurations. Each bar group corresponds to the same Q-Learning algorithm trained on the original intersection with a different normalization configuration.

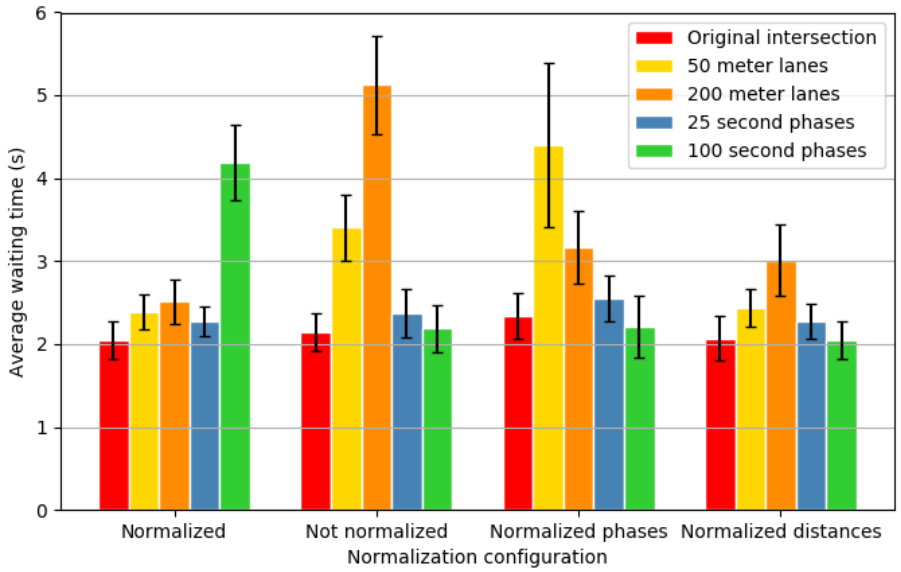


Figure 5.6: Comparison of sensitivity to different lane lengths and phase times, for different normalization configurations (original training intersection: 100 meter lanes and 50 second maximum phase time, the phase values in the legend correspond to maximum phase times).

The algorithm that only normalizes the distance to the nearest detected vehicle shows good robustness in every cases. However, the algorithm that normalizes both distance and current phase time is even less sensitive in every case, except for a longer maximum phase time. As it is supposed that the maximum phase times of an intersection on which an ITS is meant to be deployed are known or determined before training, this thesis will focus on this second algorithm (i.e. the state representation is kept as defined in section 4.2.2).

5.3.2 Traffic flow

While the lane lengths and maximum phase times of the deployment intersection may be precisely known before training, it is certainly not true for the traffic flows this intersection is subject to, that may not have the exact same average and vary a bit. Hence, even if a general quantity about these real traffic flows can be collected through historical data, robustness to traffic flow is a key characteristic for a successful deployment. Figure 5.7 shows the sensitivity to traffic flow of three algorithms trained respectively at low, medium and high traffic flow with 100% detection rate.

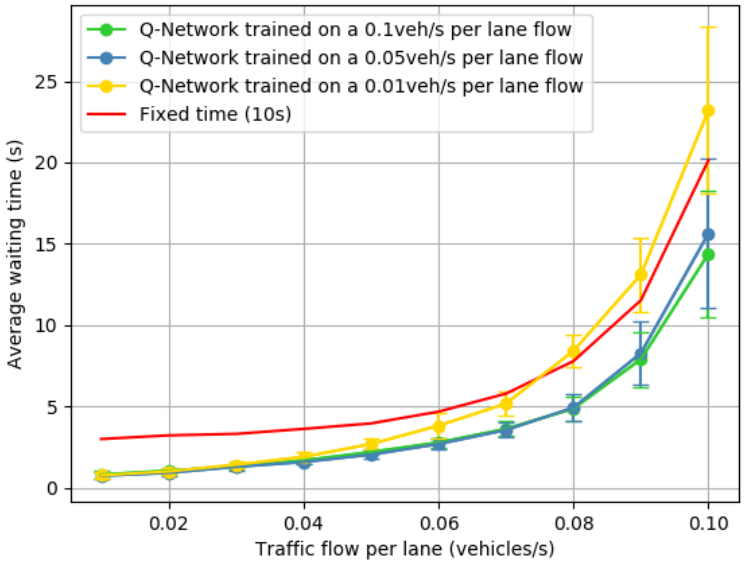


Figure 5.7: Sensitivity to traffic flow for different training flows.

It can be clearly stated from this figure that for any traffic flow an algorithm trained at high traffic flow will perform at least as well as one trained at lower traffic flow. A good piece of advice for training would thus be to overestimate the traffic flows of the intersection the ITS will be deployed on.

5.3.3 Detection rate

In the same way as for traffic flow, the detection rate at which the ITS is trained is not the same as the one at deployment time. The sensitivity of the algorithm to detection rate at medium traffic flow is pictured in Figure 5.8. It is shown that, when almost no information about the real detection rate is known, training the algorithm with a detection rate between 50% and 70% leads to good results whatever the deployment detection rate.

Overall, the tested training detection rates lead to an average waiting time that is below the fixed time baseline for any detection rate. However, this is not the case for a 100% training detection rate where the performances are getting worse and worse as the actual detection rate is lowering. This is an evidence of a different way of learning.

When reaching 100% detection rate, all vehicles passing through the intersection can be seen by the ITS, and the latter can adopt a different approach that would be catastrophic if a significant percentage of vehicles was undetectable.

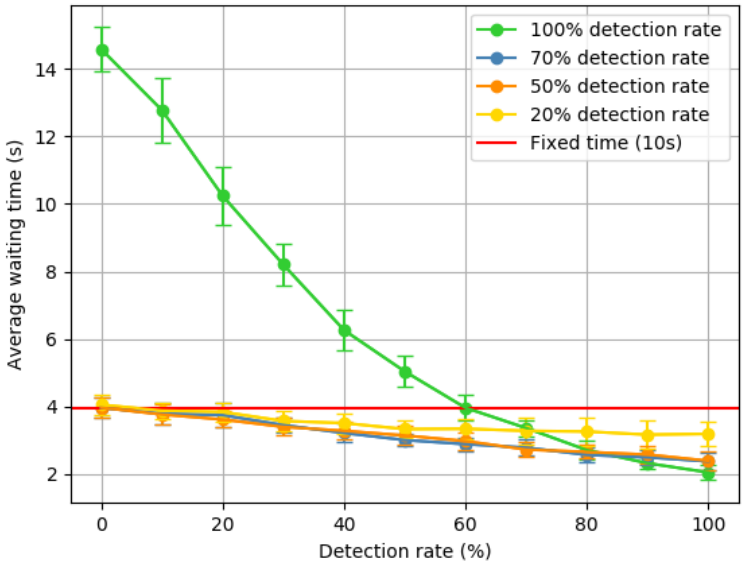
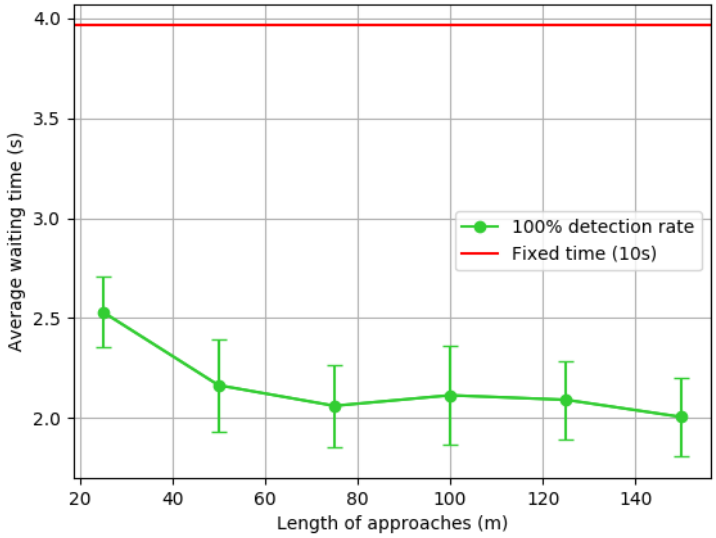


Figure 5.8: Sensitivity to detection rate for different training detection rates.

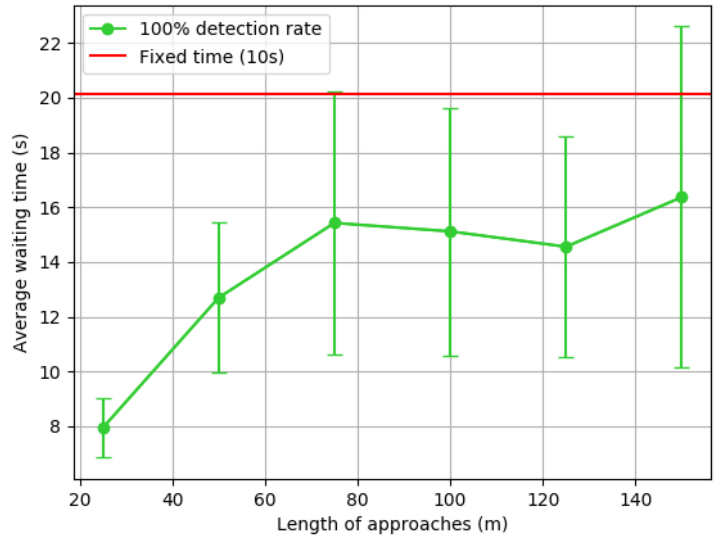
5.4 Influence of the lane length

The length of each approach of an intersection plays a huge role in the delay an ITS has to react efficiently to new vehicles. Logic would state that the further from the intersection a vehicle is detected, the better the agent will be able to adapt and the better the performances. This is not entirely true, as can be seen in Figure 5.9.

Indeed, even if Figure 5.9a confirms this hypothesis, Figure 5.9b shows that, when reaching high traffic flows, the tendency is reversed. This can be explained by the fact that, at high traffic flow, there are always vehicles to be detected on each approach and there will always be vehicles close to the intersection. In this case, what matters to take good actions is not so much the distance from the intersection but the position of the vehicles on each lane. This is why the average waiting time is significantly better for 25 meter lanes than for longer lanes in Figure 5.9b. Vehicles that are detected are grouped in the same small zone, which allows the algorithm to focus on what matters. If the lanes are longer, the need of position information for each vehicle becomes more urgent and the average waiting time increases as well as the standard deviation.



(a) Uniform medium traffic flow



(b) Uniform high traffic flow

Figure 5.9: Influence of the length of the four approaches of the simple intersection on the average waiting time.

In order to better understand the problem, an illustrated example is shown in Figure 5.10. When looking at this figure with human eyes, it can be directly seen that giving a green light to the north lane will make every car on the east lane wait and thus increase the waiting time more than necessary. However, with the state representation defined in this work and considering a 100% detection rate, the north lane will still be given a green light because more vehicles stand on it and the "Distance to the nearest detected vehicle" features are the same for north and east lanes. If the ITS had only detected cars that are inside the red square (lower lane lengths), vehicles that are far

from the intersection on the north lane would have been ignored and the algorithm would have made the right choice, i.e. letting east cars go through first.

This problem highlights a weakness of the proposed algorithm, and it would be justified to think that the algorithm might perform better when including a notion of vehicle position in each approach in the state representation, along with convolutional layers as a first forward stage of the Q-Network, for this kind of input. This scheme is not studied in this thesis but constitutes a good way of improvement for future works.

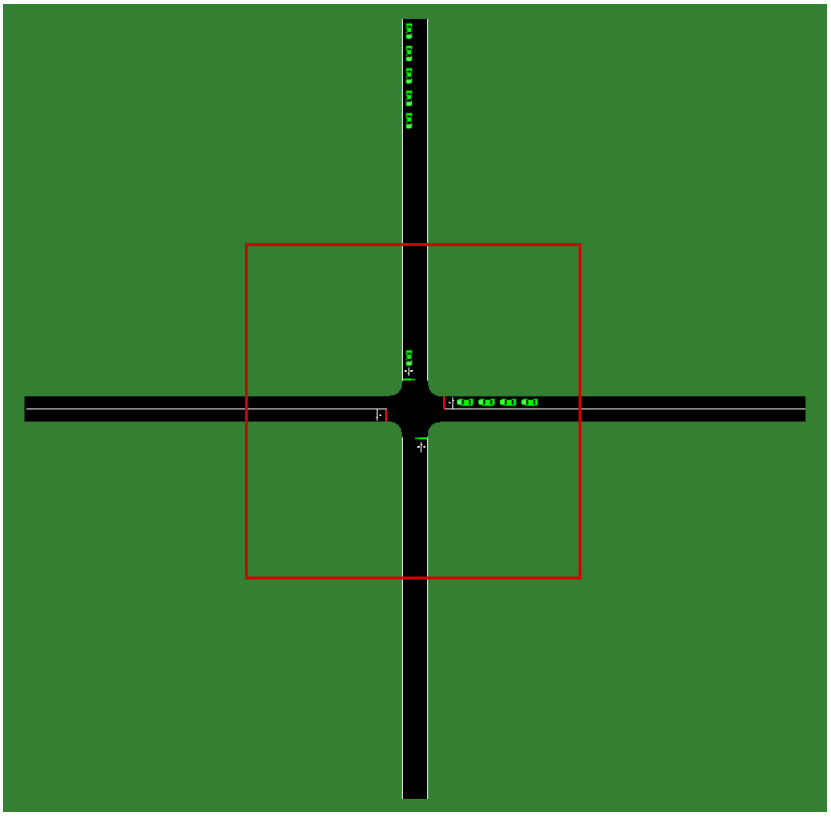


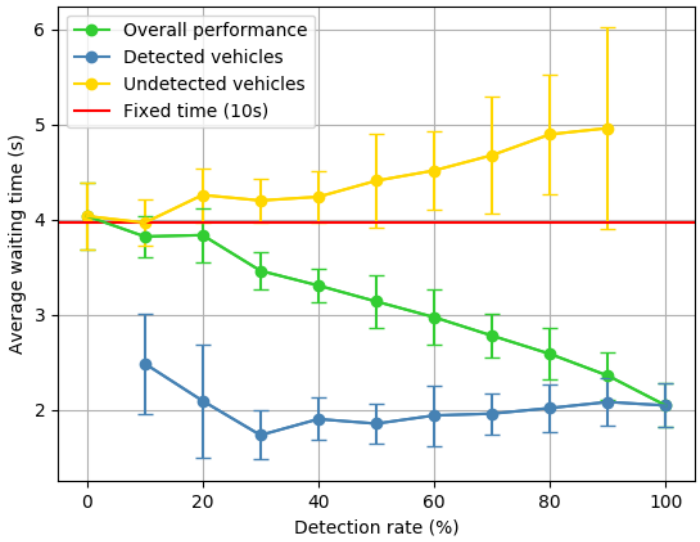
Figure 5.10: Illustrated example of the importance of vehicle position information.

5.5 Difference between detected and undetected vehicles

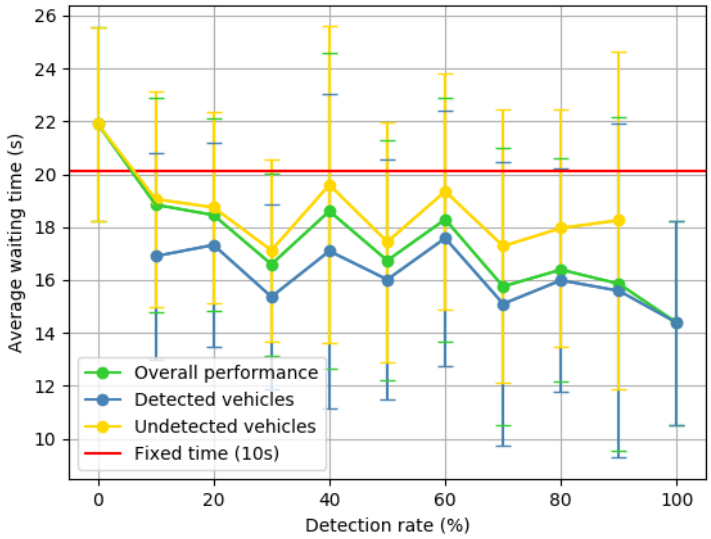
When thinking about the deployment of the ITS exposed in this work, it is obvious that, at first, a very low percentage of road-users will be equipped with the technology allowing them to be detected by the system. Therefore, it is interesting to know if anything could push road-users to switch to the technology, in order to increase detection rate and thus performances of the ITS.

Figure 5.11 shows the average waiting time of detected and undetected vehicles for different detection rates. It appears that undetected vehicles are clearly disadvantaged compared to detected vehicles, which would lead the drivers to switch to the required

technology. This shows that the solution presented in this thesis can integrate naturally into road-users' life, with possibly no cost for them depending on the technology.



(a) Uniform medium traffic flow



(b) Uniform high traffic flow

Figure 5.11: Average waiting time of detected and undetected vehicles for different detection rates.

When comparing Figures 5.11a and 5.11b, it can be seen that the difference between detected and undetected vehicles becomes tinier as the traffic flow increases. Moreover, in Figure 5.11b, the average waiting time is lower than the fixed time baseline for both detected and undetected vehicles. This is due to the fact that, when traffic flow increases, vehicles are seen more like a fluid rather than separate particles, leading to

less individualization and therefore to the possibility for undetected vehicles to leverage the advantage that is given to detected vehicles.

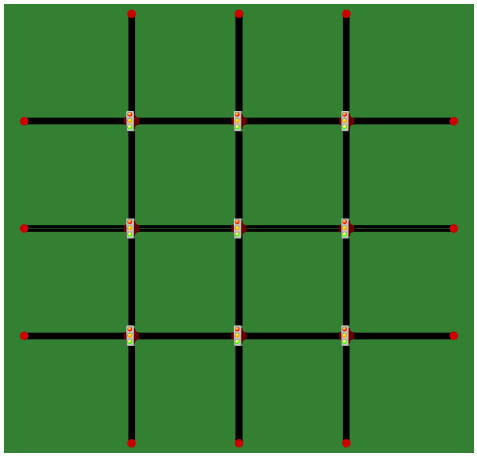
Chapter 6

Performance analysis on a road network

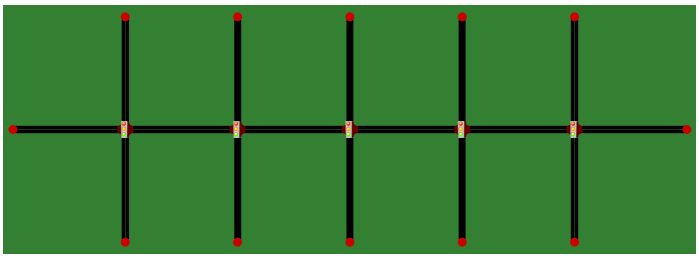
The Q-Learning algorithm performs well on a single intersection. However, when deploying such a solution in a city, it is very unlikely that it will limit to one intersection. This is why it is important to check that the ITS also provides good results on road networks, even without a multi-agent scheme (i.e. without any explicit information exchange between agents).

6.1 Description of the environments

Once again, in order to stay as general as possible, two simple and very common structures of network have been chosen as environments: a Manhattan grid and an arterial road. These two architectures are illustrated in Figure 6.1.



(a) 3×3 Manhattan grid



(b) 5×1 arterial road

Figure 6.1: Road networks used as environments.

In these networks, each intersection is the same as the one defined in section 5.1, with 100 meter lanes and the same phases as in Table 5.1. The tests are done at uniform medium traffic flow and 20%, 50%, 70% and 100% detection rates.

6.2 Independent multi-agent training

In this work, as no communication mechanisms between agents have been designed, the only way to make them adapt better to the environment than just deploying several instances of a same pre-trained agent is independent multi-agent training.

Independent multi-agent training consists in training directly on a road network as many independent agents as there are intersections. Even if no explicit information is shared between agents, training them together in the same simulated episodes enables each agent to learn how to adapt to the others. Indeed, the actions of an agent that lets vehicles pass on a particular road are reflected in the incoming traffic flow of the intersection located at the other end of the road and controlled by another independent agent.

6.3 Performances

The road networks allow us to test two aspects of the algorithm:

1. The performances of a single agent trained on a simple intersection and deployed on every intersection of the network.

2. The performances of multiple agents, each corresponding to one intersection of the network, directly trained together and deployed on the network, as described in section 6.2.

These two separate tests enable to assess:

1. The possibility to deploy a general single agent on several intersections with good performances.

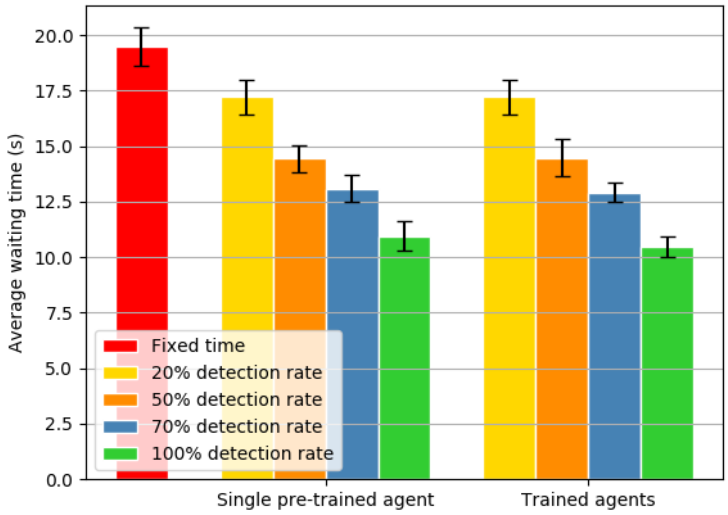
2. How much improvement is brought by training several agents, each one specialized for a specific intersection.

The performances in terms of waiting time of the single pre-trained agent and the specialized agents compared to fixed time control are presented in Figure 6.2.

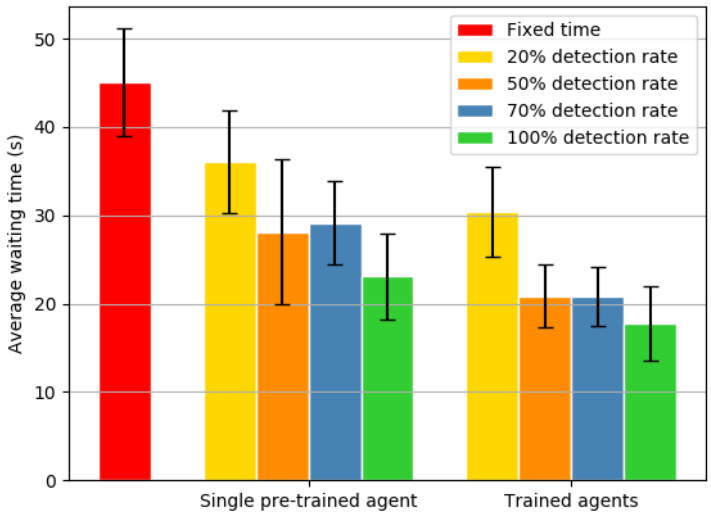
The results for the Manhattan grid in Figure 6.2a display no difference between the agent trained on a single intersection and the specialized agents. This is due to the symmetry of the network. Indeed, as each entry of the network is subjected to the same medium traffic flow, the traffic flows in every lane are thus the same and a single agent trained at medium traffic flow on a simple intersection fits perfectly to every intersection of the network. The only difference between a single simple intersection and an intersection of the Manhattan grid is that the latter is subjected to more realistic, non-continuous traffic flows. This, however, does not seem to affect performances.

In Figure 6.2b are shown the results for the arterial road. The situation is different in this case as, even though they are the same in each entry, the traffic flows are different in each internal lane of the artery. This asymmetry in traffic flow leads to

the issue that a single agent cannot be adapted exactly to each and every intersection, leading to better performances for multiple agents trained simultaneously, directly on the network.



(a) 3×3 Manhattan grid



(b) 5×1 arterial road

Figure 6.2: Performances of a single pre-trained agent and trained agents on the two road networks.

In conclusion, in order to obtain less congestion and waiting time for vehicles, the best solution consists in training multiple agents together on a simulated road network corresponding to the deployment map. Training a single agent for each separate intersection could also be considered rather than taking a more general pre-trained agent.

It is however worth mentioning that both approaches described in this section provide good performances compared to a fixed time system.

Chapter 7

Deployment on a real intersection

Although sensitivity and adaptability tests are essential to the validation process of a control system like the one described in this work, it is also very important to prove that the ITS will deliver good results on a real deployment case, with training environment and data that are simple enough to generalize the proof to any other deployment case. This chapter follows this purpose.

7.1 Description of the environment

7.1.1 The LuST scenario

The intersection used for deployment is part of a scenario called the Luxembourg SUMO Traffic (LuST) scenario [25, 26]. It was designed by Codecá et al. in order to provide a scenario that meets the requirements of scientists who want to test their ITS on reproduced, realistic mobility patterns. The city of Luxembourg was mostly chosen because the government provides traffic statistics that enable to calibrate traffic demand. Moreover, this city has a reasonable size with respect to a microscopic simulator like SUMO and has a topology that is complete enough and comparable to many European cities. This traffic data is provided on the website of the Luxembourg National Institute of Statistics and Economic studies (STATEC) [27].

In the scope of the algorithm presented here, this scenario is interesting because it takes place over a 24 hour period, which exploits the "time of the day" feature in the state representation. Moreover, despite the fact that the pedestrians and some specific reserved lanes are not simulated, which would have added more realism, it provides a sufficiently realistic traffic flow based on real historical traffic data. The scenario simulates a deterministic traffic demand in the whole city of Luxembourg, with rush hour peaks at 8:00 and 18:30 and an off-peak around 13:00.

7.1.2 Training and deployment intersections

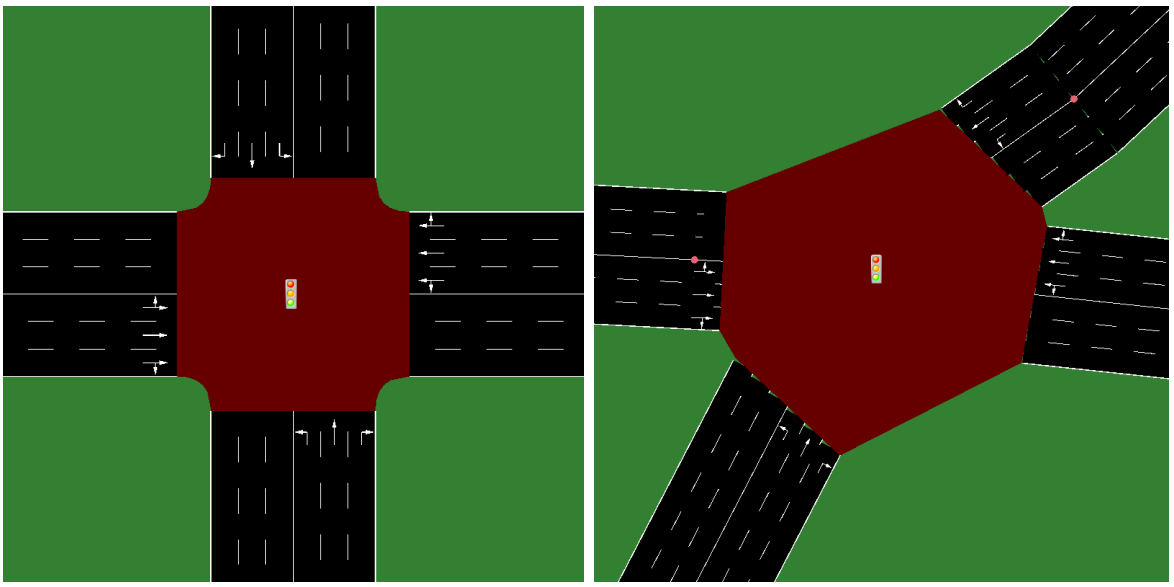
So far, the intersections on which the algorithm was tested were simple and general. The goal here is not only to test the ITS against real challenging traffic flows but also to deploy it on an intersection that is realistically complex. In the LuST scenario, the

junction identified by −12408 is a very frequented intersection providing enough complexity (several lanes per incoming road, specific directions for each lane, more light phases, ...).

In order to demonstrate the robustness of the algorithm and the generalization of the training process, the training intersection is chosen to be a simplified replica of the real intersection with simplified traffic patterns. The whole deployment process, which can also be applied on real physical intersections rather than a scenario, consists of:

1. Measuring, lane by lane, the hourly traffic volumes of the real intersection.
2. Creating a simplified version of the real environment, with traffic flows corresponding to these real volumes.
3. Training the RL agent on this newly created environment.
4. Substituting the newly trained agent to the fixed time system on the real intersection.

Figure 7.1 shows the training and deployment LuST intersections. Note that the traffic light phases for these intersections are exactly the same and are defined in appendix C. Table 7.1 highlights the differences between the two intersections. The hourly traffic volume measured on the deployment intersection and simulated on the training intersection is also presented in Figure 7.2. The rush hours and the off-peak around lunch time are clearly visible.



(a) Training intersection (b) Deployment intersection

Figure 7.1: Topology of the training and deployment (in LuST) intersections.

	Training	Deployment (LuST)
Lane length	100 meters	Equal to the length of the shortest approach[1]
Traffic flow generation	Bernoulli distribution	Bulk arrival
Type of vehicle	Cars	Cars and buses
Stop-and-go vehicles	None	Bus stops
Vehicle speed	Constant	Gaussian mixture distribution
Vehicle passing	None	Passing due to speed randomness

Table 7.1: Differences between the training intersection and the deployment intersection from the LuST scenario.

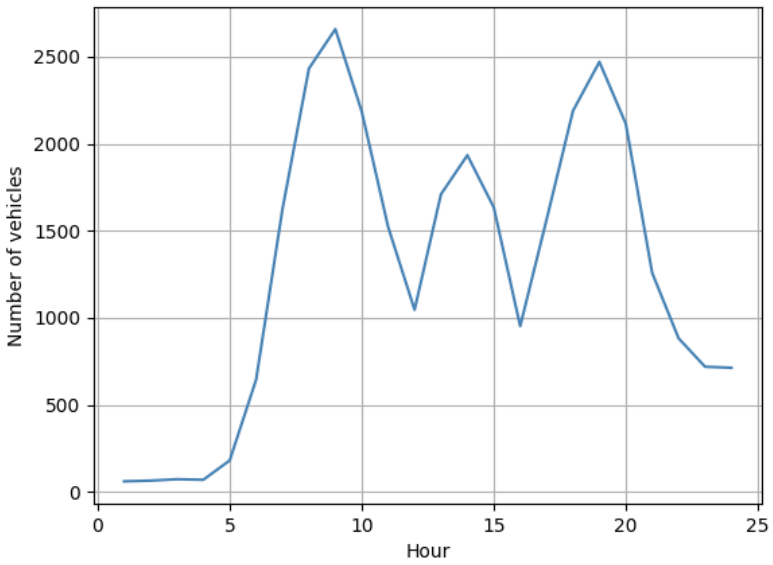


Figure 7.2: Hourly traffic volume on the deployment intersection from the LuST scenario.

7.2 Performances

As an evaluation of the deployment case, Figure 7.3 shows the hourly performances of the algorithm on the real intersection from the LuST scenario, in terms of waiting time. Error bars are not displayed for the simple reason that the LuST scenario relies on a deterministic traffic generation. It is clear that the algorithm provides a lower

[1]This is necessary when the "Distance to the nearest detected vehicle" feature is normalized in order to keep comparison at scale between lanes. Hence, it brings out a limitation of this approach. Indeed, not normalizing would allow different lane lengths and thus to consider each entire incoming road.

average waiting time than fixed time control, at any hour of the day, including and most importantly at rush hours, and this even at low detection rate.

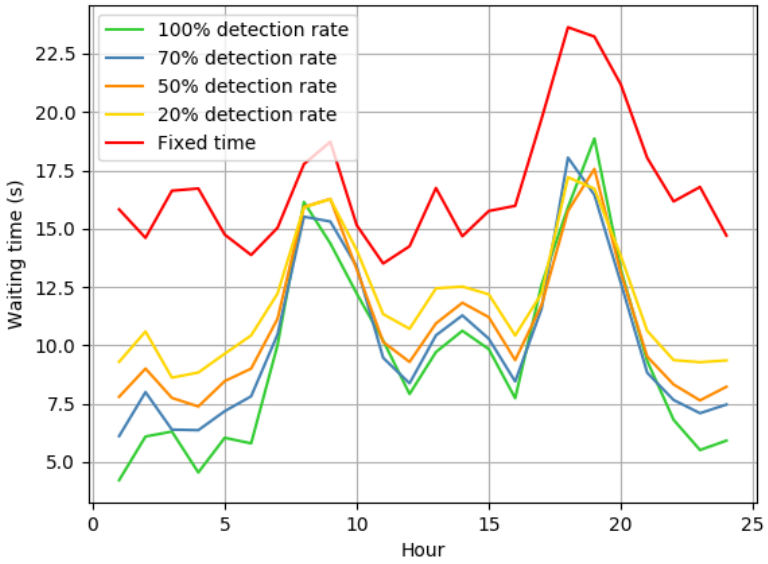


Figure 7.3: Hourly waiting time on the LuST deployment intersection, for different detection rates.

It can also be noticed that the difference between the detection rates in terms of waiting time decreases as the traffic volume increases, with roughly the same performances for all the detection rates at rush hours. As explained in section 5.5, this is due to the fact that the higher the traffic flow, the more traffic is seen as a fluid rather than a collection of particles (vehicles), and thus the more undetected vehicles can take advantage of the detection of other vehicles coming from the same lane. This phenomenon is accentuated with the realistic mobility patterns of the LuST scenario, as vehicles are arriving to the intersection by waves.

Chapter 8

Integration of a priority system

In this chapter, a priority system for public transports, particularly buses, is designed and tested. The integration of such a system into the original ITS could encourage people to take public transports, with three benefits:

- An obvious ecological benefit.
- The more people in public transports, the lower the average waiting time per road-user (not per vehicle).
- Since all public transports should be detected by the ITS, more people in public transport would mean less people in possibly undetected vehicles, and thus an increase of the detection rate and a decrease of the congestion level, leading to better performances.

8.1 Methodology and scenario

Methodology The integration of the priority system into the algorithm is straightforward. It consists in separating buses and other vehicles in the feature vector given as input to the Q-Network, in order for the agent to differentiate the two types. For this purpose, the entries for the "Number of detected vehicles" feature are doubled. Also, the reward corresponding to a bus is multiplied by a priority factor $p \geq 1$ such that:

$$r_t^{bus} = p.\frac{v_{ITS}^{bus}(t) - v_{max}^{bus}}{v_{max}^{bus}} \ . \tag{8.1}$$

This introduces a bigger penalty if a bus is slowed down or kept waiting rather than another vehicle.

Scenario The scenario takes place on the simple intersection described in section 5.1, with the exact same phases. For challenging tests, this intersection is meant to be frequently crossed by buses. This is why three two-way bus lines (see characteristics listed in Table 8.1) are passing through the intersection. These bus lines are only representing paths to follow within the intersection and are not to be confused with special lanes for buses.

	Path	**Passage frequency**	**Difference between planned and actual passage time**
Bus line 1	East lane to west lane and back	Every $10'$	Gaussian distribution with a standard deviation of $1'30''$
Bus line 2	North lane to east lane and back	Every $15'$	
Bus line 3	North lane to south lane and back	Every $10'$	

Table 8.1: Characteristics of the bus lines.

8.2 Results

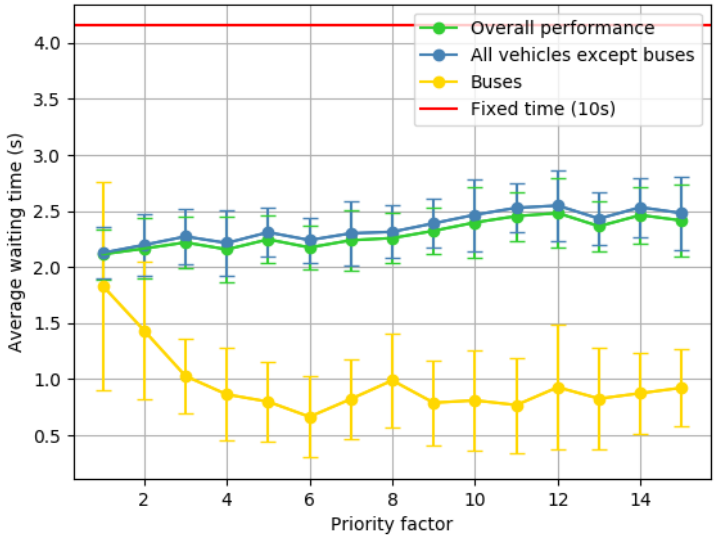
The goal is to find the right value for the priority factor, so that buses are significantly favored without unnecessarily penalizing the rest of the vehicles[1]. Figures 8.1 and 8.2 show the influence of the priority factor on the average waiting time, separating buses from the rest of the vehicles. By looking at the charts, it seems obvious that it is not necessary to go over a priority factor of 4, as the waiting time of buses is not reduced anymore and the waiting time of other vehicles is even slightly increased.

Comparing Figures 8.1a and 8.1b, it is quite visible that the curve corresponding to buses is not starting at the same average waiting time as the others at 20% detection rate. This is due to the simple fact that buses are considered to be fully detected. Therefore, even for a priority factor of 1 (no prioritization), buses already benefit from always being seen by the agent. Another interesting point is that, as the priority factor increases, the average waiting time of buses seems to converge faster to its optimal value at lower detection rates. Indeed, if less cars are detected, the proportion of buses among detected vehicles increases. This means less competition for buses. Therefore a smaller priority factor is enough to make the difference.

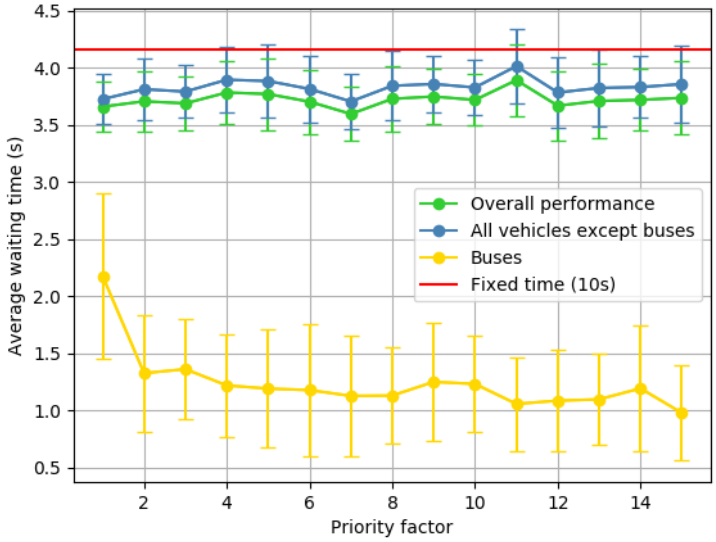
A lower detection rate also means worse performances for other vehicles than buses. This bad performance seems to slightly rub off on that of buses since the optimal average waiting time of buses is lower than 1 in Figure 8.1a, at 100% detection rate, and higher than 1 in Figure 8.1b, at 20% detection rate.

Now taking an interest in the effect of traffic flow, Figure 8.2b displays no differences in the trends at high traffic flow compared to medium traffic flow. Moreover, Figure 8.2a shows that, at low traffic flow, the average waiting time for buses is constant. Hence, low traffic flows simplify the problem enough, so that the agent is able to optimize the average waiting time of every vehicle without the need of prioritization.

[1]Another idea would be to make use of a different priority factor for each bus, proportional to the number of passengers it carries. This approach optimizes the waiting time for each road-user. However, in this work, from an ecological point of view, it has been chosen to favor buses as much as possible in order to encourage people to take public transports.

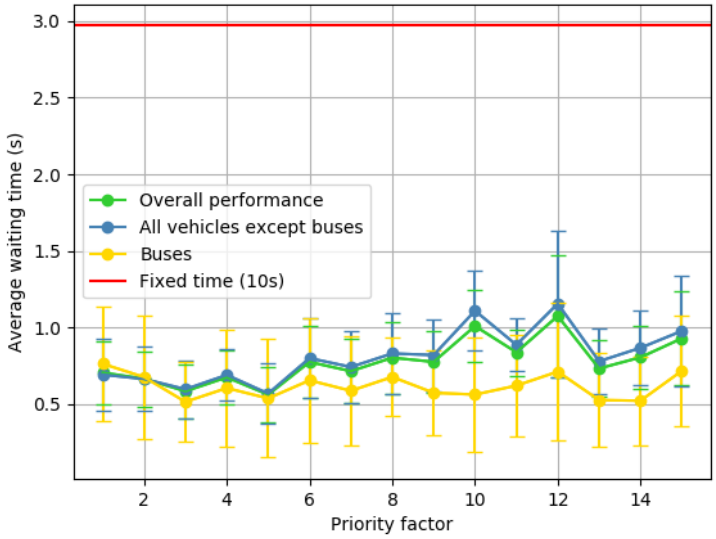


(a) 100% detection rate

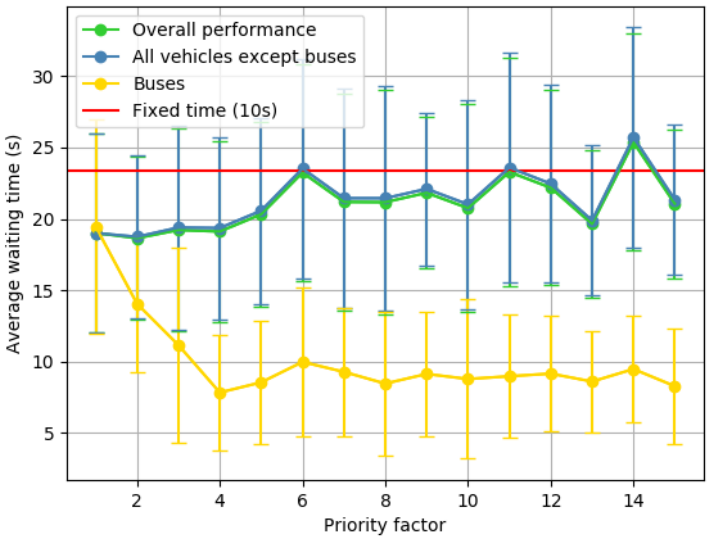


(b) 20% detection rate

Figure 8.1: Influence of the priority factor on the average waiting time at uniform medium traffic flow, separating buses and other vehicles.



(a) Uniform low traffic flow



(b) Uniform high traffic flow

Figure 8.2: Influence of the priority factor on the average waiting time with a 100% detection rate, separating buses and other vehicles.

Chapter 9

Integration of pedestrians

An important next step in the process of solving the congestion problem at traffic light intersections is to take pedestrians into account at relevant intersections. As a matter of fact, not only would considering pedestrians in intersections featuring pedestrians reduce the waiting time for them, but, if a good scheme is to be found, it could also improve the performances for vehicles: the environment and its corresponding state observations would get closer to reality. This chapter analyzes several approaches focused on pedestrians and incorporable into the original algorithm.

9.1 Definition of the environment

The intersection on which the different algorithms are trained and tested is the same as the simple intersection described in section 5.1, except that:

- Sidewalks and four pedestrian crossings are added.
- For the sake of safety and realism, the traffic light phases are changed for both vehicles and pedestrians.

Figure 9.1 displays this intersection and Table 9.1 explains the different light phases. The pedestrian flow configurations are exactly the same as the traffic flow configurations defined in Table 5.2.

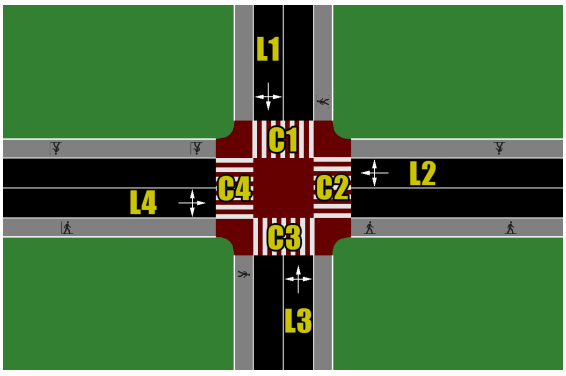


Figure 9.1: Intersection featuring pedestrians on which the algorithms are tested (L=lane, C=crossing).

	L1	L2	L3	L4	C1	C2	C3	C4	Duration
Phase 1	G	R	G	R	R	G	R	G	Minimum: 10″ Maximum: 50″
Phase 2	G	R	G	R	R	R	R	R	5″
Phase 3	A	R	A	R	R	R	R	R	3″
Phase 4	R	G	R	G	G	R	G	R	Minimum: 10″ Maximum: 50″
Phase 5	R	G	R	G	R	R	R	R	5″
Phase 6	R	A	R	A	R	R	R	R	3″

Table 9.1: Description of the 6 phases of the intersection featuring pedestrians (L=lane, C=crossing, colors correspond to light colors: G=green, R=red, A=amber).

9.2 Description of the algorithms

The algorithms described in this section consider pedestrians as vehicles, moving in different lanes (sidewalks) than car lanes. Therefore, the same values are associated to vehicles and pedestrians, but as different entries of the Q-Network (different state values). In order to better understand the idea behind each algorithm, the descriptions are based on the illustration presented in Figure 9.2.

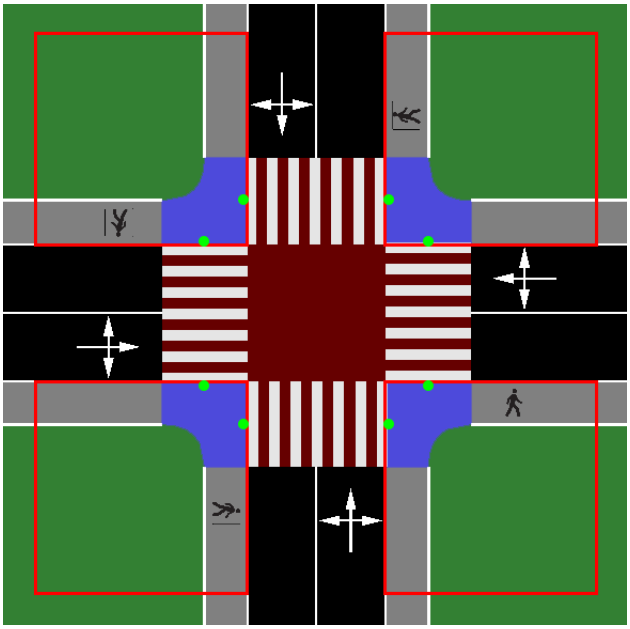


Figure 9.2: Supporting illustration for the description of the pedestrian algorithm ideas.

The algorithms described below exploit two main approaches: separating pedestrians either into 4 zones demarcated by the crossroads (algorithms identified by "Zone$N$") or by crossing (algorithms identified by "Crossing$N$").

1. **Zone1**: counts the number of pedestrians waiting in each blue zone (4 "Number of detected pedestrians" entries).

2. **Zone2**: same as Zone1, but also counts pedestrians moving in the red squares (4 "Number of detected pedestrians" entries).

3. **Zone3**: same principle as for vehicles. In each zone, counts the number of pedestrians coming towards the intersection and takes the distance between the intersection and the nearest pedestrian, considering this time the entire sidewalks (4 "Number of detected pedestrians" entries, 4 "Distance to the nearest detected pedestrian" entries).

4. **Crossing1**: counts the number of pedestrians waiting at each crossing, in each blue zone. Separation between crossings in each blue zone is done by choosing the crossing whose green dot is closest to the pedestrian (4 "Number of detected pedestrians" entries).

5. **Crossing2**: same as Crossing1, but also counts pedestrians moving in the red squares (4 "Number of detected pedestrians" entries).

6. **Crossing3**: same principle as for vehicles. Separating by crossing, counts the number of pedestrians that are coming towards the intersection and takes the distance between the intersection and the nearest pedestrian, considering this time the entire sidewalks. Separation between crossings in each zone is done by choosing the crossing whose green dot is closest to the pedestrian (4 "Number of detected pedestrians" entries, 4 "Distance to the nearest detected pedestrian" entries).

In order to avoid penalties regarding pedestrians voluntarily stopping on their way, the rewards and waiting times are calculated only considering pedestrians standing in the blue zones. Therefore, the only misleading case is the one where a pedestrian is voluntarily stopping inside a blue zone, without the wish to cross the road (this would however never happen in simulation).

9.3 Results

The performances of the 6 algorithms described in the previous section are compared with those of:

- A fixed time control system ("Fixed" in the figures).
- The original algorithm, that neglects pedestrians ("Ped. neglected" in the figures).
- An algorithm that relies on the same principle as Crossing3 and has perfect information about the trajectory of every pedestrian, allowing it to provide the exact correct values as entries of the Q-Network ("Perfect" in the figures).

Figure 9.3 shows the average waiting time for the different algorithms at uniform medium traffic and pedestrian flows, separating vehicles and pedestrians. The performances of all the algorithms are quite identical, even in a perfect information case, leading to the conclusion that no improvement can be made in this configuration. This is due to two very limiting factors:

1. By adding pedestrians and sidewalks, the number of independent, incoming approaches to the intersection and the total flow are doubled, while the agent still has the same number of alternatives (either let vehicles and pedestrians pass from North to South and back, or from West to East and back). This limits the chances to find an alternative that suits well to the current situation.

2. Safety requires two additional traffic light phases to be added, corresponding to phases 2 and 5 in Table 9.1 that show a red light to all pedestrians. This means that, while a transition from a vehicle green light phase to another still lasts 3 seconds, a transition from a pedestrian green light phase to another lasts up to 8 seconds. This highly reduces the light maneuverability for pedestrians. Moreover, vehicle and pedestrian transitions depending on each other, bad performances regarding pedestrians somehow rub off on performances regarding vehicles.

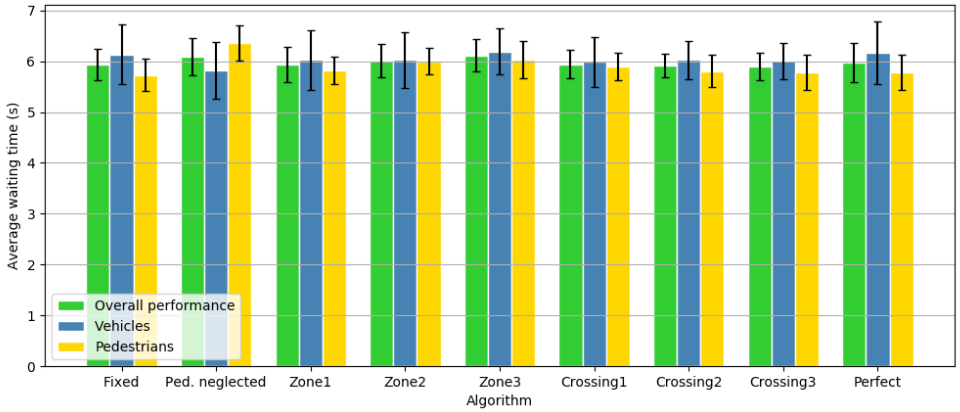


Figure 9.3: Average waiting time at uniform medium traffic and pedestrian flows, with 100% detection rate.

Knowing these limiting factors, in order to be able to compare the different algorithms, two actions are taken:

1. The vehicle and pedestrian flows are reduced to low.

2. The length of all the lanes is increased to 300 meters instead of 100 meters. This allows for more reaction time for the agent.

Figure 9.4 shows the corresponding results. The most important point is that performances for fixed time control and for perfect information are still very close and with overlapping error bars, meaning that not much improvement can be achieved with the simple feature RL algorithm proposed in this thesis.

By looking at the average waiting times concerning the original algorithm (ped. neglected in Figure 9.4), the usefulness of taking pedestrians into account in intelligent traffic light control is verified, especially since the average waiting time for vehicles is not even better than the ones of the other algorithms.

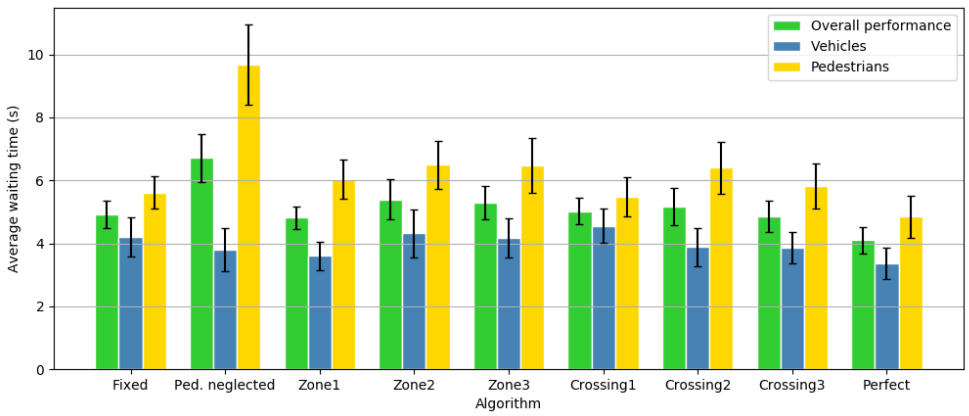


Figure 9.4: Average waiting time at uniform low traffic and pedestrian flows, with 100% detection rate and 300 meter lanes.

The two algorithms that seem to bring the best performances are Zone1 and Crossing3, i.e. respectively the simplest and most sophisticated ones. Moreover, these performances are similar to those of a fixed time system. This shows once again that the problem of pedestrian integration cannot be efficiently solved or is at least too complex for the state representation defined in this work.

Chapter 10

Conclusion

This work aimed to design a Connectionist Reinforcement Learning algorithm based on Vehicle to Infrastructure (V2I) communications and was therefore dealing with partial detection. The main purpose was to perform an efficient traffic light control, even at low detection rate, and to reduce the waiting time of every road-user. Overall, the resulting algorithm showed significantly better performances than currently installed fixed time control systems.

The present thesis was based on an article from Zhang et al. [5]. The methodology and a reassessed performance analysis were presented in a more detailed and transparent way. In addition, focusing on the optimization of the waiting time of every road-user, the integration of pedestrians and of a priority system for public transports were studied. These aspects of traffic light control are quite new to the scientific literature.

Results demonstrated that the proposed Intelligent Transport System was able to reduce congestion, even at low detection rates, and was robust enough with respect to different parameters (traffic flow, detection rate,...) to adapt to different situations and to be effectively deployed on a real intersection.

It was also shown that prioritization of buses or other public transports is efficient with the algorithm and constitutes a good idea in many ways. However, the study of the integration of pedestrians led to the conclusion that traffic light control at intersections featuring pedestrians is a completely new challenge that would at least require as a working basis a more complex solution than the one introduced in this work. Indeed, the state representation was too simple to overcome the difficulty of managing pedestrians and vehicles at the same time.

The difference in waiting time between detected and undetected vehicles was verified. This difference can lead to an easier transition to higher detection rates, and thus better performances. For example, one could think of advertisements for the new V2I technology (required by the algorithm) that highlights the time-saving aspect of the product.

10.1 Future work

The lack of stability of the algorithm at high traffic flow as well as the analysis of the influence of the lane length showed that taking the position of each vehicle on the lanes into account, possibly adding convolutional layers in the Q-Network, may lead to even less congestion. This is definitely a path to explore, especially since this more sophisticated approach could demonstrate better performances on intersections featuring pedestrians.

Since the RL agent is not meant to stand at only one intersection but is more likely to be deployed as several instances on a road network, it is important to develop a multi-agent scheme. Indeed, in this kind of configuration, deploying several independent agents is suboptimal and establishing efficient communication between them could greatly reduce commute time of road-users.

The work of Zhang et al. about the environmental adaptation of their algorithm [18] outlines two interesting points:

- Policy-based algorithms can adapt to changing environments more efficiently than value-based algorithms.
- The different algorithms led to satisfying results using a partial reward (calculation based on detected vehicles) instead of a full reward (calculation based on all vehicles, as used in this thesis).

Continuing the work with a policy-based agent and a partial reward might therefore be a good idea. The latter would enable the agent to continue to train once deployed.

Given the available historical traffic data, the performances of the presented ITS were assessed only on periods of at most 24 hours. With larger data, it would be interesting to add a "Day of the week" feature, in order to differentiate high traffic flow days from low traffic flow days and possibly further reduce congestion.

References

[1] D. I. Robertson. "'transyt' method for area traffic control". In: 1969.

[2] Peter W. Hunt et al. "The scoot on-line traffic signal optimisation technique". In: 1982.

[3] Pamela B. Lowrie. "Scats, Sydney co-ordinated adaptive traffic system: a traffic responsive method of controlling urban traffic". In: 1990.

[4] Stephen Smith et al. "SURTRAC: Scalable Urban Traffic Control". In: *Proceedings of Transportation Research Board 92nd Annual Meeting Compendium of Papers*. Transportation Research Board, Jan. 2013.

[5] Rusheng Zhang et al. "Partially Observable Reinforcement Learning for Intelligent Transportation Systems". In: *CoRR* abs/1807.01628 (2018). arXiv: 1807.01628. URL: http://arxiv.org/abs/1807.01628.

[6] J. B. Kenney. "Dedicated Short-Range Communications (DSRC) Standards in the United States". In: *Proceedings of the IEEE* 99.7 (2011), pp. 1162–1182.

[7] Rusheng Zhang et al. "Virtual Traffic Lights: System Design and Implementation". In: *CoRR* abs/1807.01633 (2018). arXiv: 1807.01633. URL: http://arxiv.org/abs/1807.01633.

[8] S. Mikami and Y. Kakazu. "Genetic reinforcement learning for cooperative traffic signal control". In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. 1994, 223–228 vol.1.

[9] Marco Wiering. "Multi-Agent Reinforcement Learning for Traffic Light Control." In: Jan. 2000, pp. 1151–1158.

[10] Marco Wiering et al. "Simulation and optimization of traffic in a city". In: July 2004, pp. 453–458. ISBN: 0-7803-8310-9. DOI: 10.1109/IVS.2004.1336426.

[11] Ella Bingham. "Reinforcement learning in neurofuzzy traffic signal control". In: *European Journal of Operational Research* 131.2 (2001). Artificial Intelligence on Transportation Systems and Science, pp. 232–241. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(00)00123-5. URL: http://www.sciencedirect.com/science/article/pii/S0377221700001235.

[12] I. Arel et al. "Reinforcement learning-based multi-agent system for network traffic signal control". In: *IET Intelligent Transport Systems* 4.2 (2010), pp. 128–135.

[13] R. Wunderlich et al. "A Novel Signal-Scheduling Algorithm With Quality-of-Service Provisioning for an Isolated Intersection". In: *IEEE Transactions on Intelligent Transportation Systems* 9.3 (2008), pp. 536–547.

[14] Xiaoqiang Wang et al. *Large-scale Traffic Signal Control Using a Novel Multi-Agent Reinforcement Learning*. 2019. arXiv: 1908.03761 [cs.LG].

[15] Elise Van der Pol and Frans A. Oliehoek. "Coordinated Deep Reinforcement Learners for Traffic Light Control". In: *NIPS'16 Workshop on Learning, Inference and Control of Multi-Agent Systems*. Dec. 2016. URL: https://sites.google.com/site/malicnips2016/papers.

[16] Juntao Gao et al. "Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network". In: *CoRR* abs/1705.02755 (2017). arXiv: 1705.02755. URL: http://arxiv.org/abs/1705.02755.

[17] Tianshu Chu et al. "Multi-Agent Deep Reinforcement Learning for Large-scale Traffic Signal Control". In: *CoRR* abs/1903.04527 (2019). arXiv: 1903.04527. URL: http://arxiv.org/abs/1903.04527.

[18] Rusheng Zhang et al. *Partially Detected Intelligent Traffic Signal Control: Environmental Adaptation*. 2019. arXiv: 1910.10808 [eess.SP].

[19] Gilles Louppe. "Deep Learning". Lecture 2: Neural networks. URL: https://github.com/glouppe/info8010-deep-learning.

[20] Gilles Louppe. "Introduction to Artificial Intelligence". Lecture 8: Making decisions. URL: https://github.com/glouppe/info8006-introduction-to-ai.

[21] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning*. 1992, pp. 279–292.

[22] Daniel Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements*. International Journal On Advances in Systems and Measurements 5.3&4 (Dec. 2012), pp. 128–138. URL: http://elib.dlr.de/80483/.

[23] *Traffic flows used in the tests*. 2019. URL: https://www.youtube.com/watch?v=M38z9SezQIU (visited on 05/20/2019).

[24] *Performances of the Intelligent Transport System on a simple intersection*. 2019. URL: https://www.youtube.com/watch?v=FeOBd9bSUoM (visited on 05/20/2019).

[25] L. Codeca, R. Frank, and T. Engel. "Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research". In: *2015 IEEE Vehicular Networking Conference (VNC)*. 2015, pp. 1–8.

[26] Lara Codecá et al. "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation". In: *IEEE Intelligent Transportation Systems Magazine* 9.2 (2017), pp. 52–63.

[27] *Luxembourg National Institute of Statistics and Economic studies website*. URL: https://www.statistiques.public.lu (visited on 05/26/2019).

Appendix A

Architecture and hyperparameters of the Q-Network

Architecture The Q-Network used in this work, as well as its corresponding target network, is composed of two fully connected hidden layers of 512 nodes each, an input layer representing a state observation whose number of nodes varies with the environment, and an output layer of 2 nodes that give the Q-values for the possible actions: either stay at the current phase or switch to the next one. Rectified Linear Unit (ReLU) activation functions are also used in each layer.

Hyperparameters The hyperparameters used for training are listed in Table A.1.

Number of iterations per episode	3000
Replay buffer	Capacity: 100000 Initialized with 10000 transitions generated with a random policy
Loss function	Mean Squared Error (MSE)
Optimizer	Adam
Scheduled learning rate α	0.0001 at episodes $[0, 49]$ 0.00001 at episodes $[50, 99]$ 0.000001 at episodes ≥ 100
ϵ (ϵ-greedy policy)	Initially 1, decays linearly to 0.05 in 100000 iterations
Discount factor γ	0.9
Batch size	32
Target network update frequency	Every 3000 iterations

Table A.1: Hyperparameters of the Q-Network training process.

Training is done over a certain number of episodes, each one lasting for a certain number of iterations, with an ϵ-greedy policy. This means that, at each iteration, the agent takes a random action with probability ϵ or takes the action corresponding to

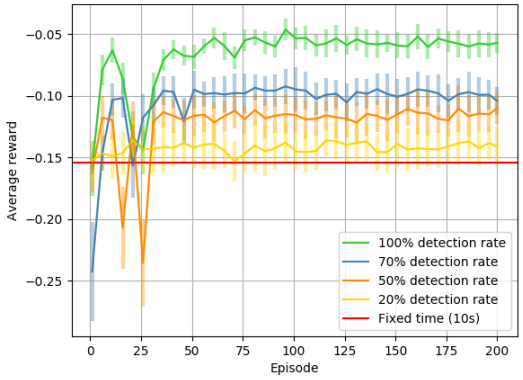
the highest Q-value with probability $1 - \epsilon$.

The agent usually trains on episodes of 3000 iterations, which corresponds to 3000 seconds in the SUMO simulator (see section 4.3). However, when it is required to train on episodes that correspond to days, the number of iterations per episode is 86400 and the capacity and initial number of transitions of the replay buffer are adapted to 3000000 and 300000 respectively.

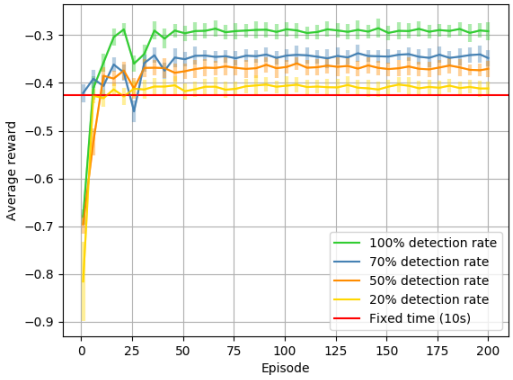
Appendix B

Training curves of the algorithm on the simple intersection

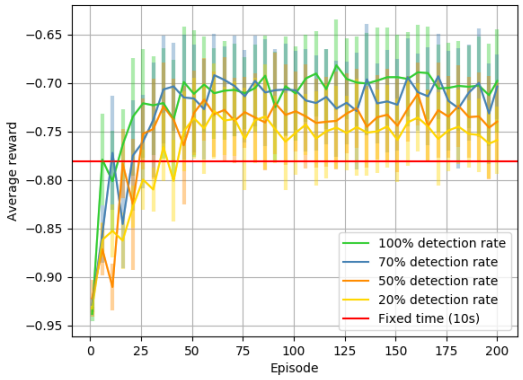
For information only, this appendix displays in Figures B.1 and B.2 the training curves of the agent over all the episodes, for all the traffic flow configurations presented in section 5.1.



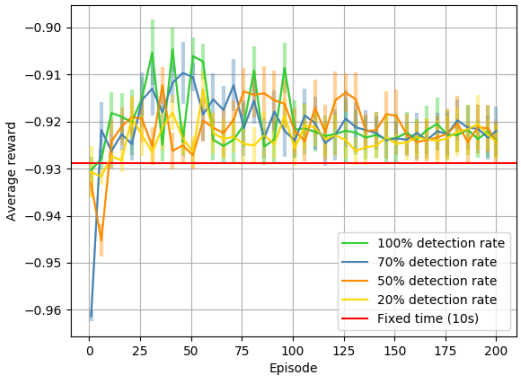
(a) Low



(b) Medium



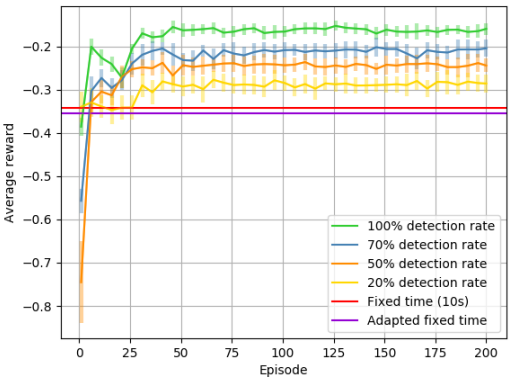
(c) High



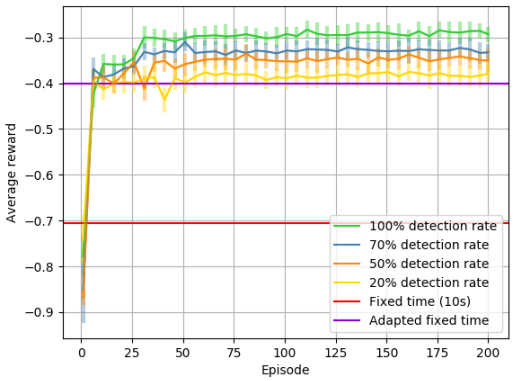
(d) Very high

Figure B.1: Uniform traffic flow

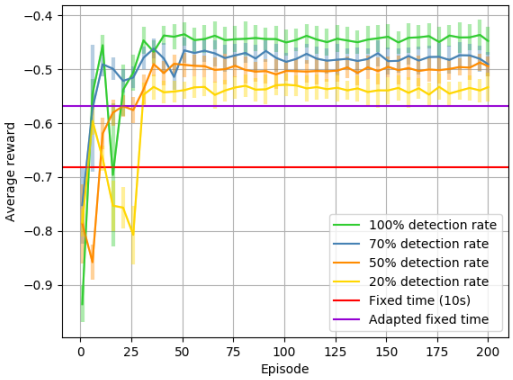
In Figure B.2, the "Adapted fixed time" purple line corresponds to adapted fixed time phases whose duration is proportional to traffic flow, as if the traffic flow in each lane was exactly known, which is an ideal case.



(a) Medium - Low



(b) High - Low



(c) High - Medium

Figure B.2: Non-uniform traffic flow

Appendix C

Traffic light phases of the training and deployment intersections

The training intersection for the LuST deployment case is represented in Figure C.1. The identifier "RN" (for "road segment") corresponds to the light that is shown to all vehicles on all the lanes of a road segment, whatever their direction, except the ones that are turning left. The identifier "LN" (for "left turn") corresponds to the light that is shown to the vehicles of the leftmost lane of a road segment that are turning left. The exact same logic can be followed with the topology of the LuST deployment intersection in Figure 7.1b.

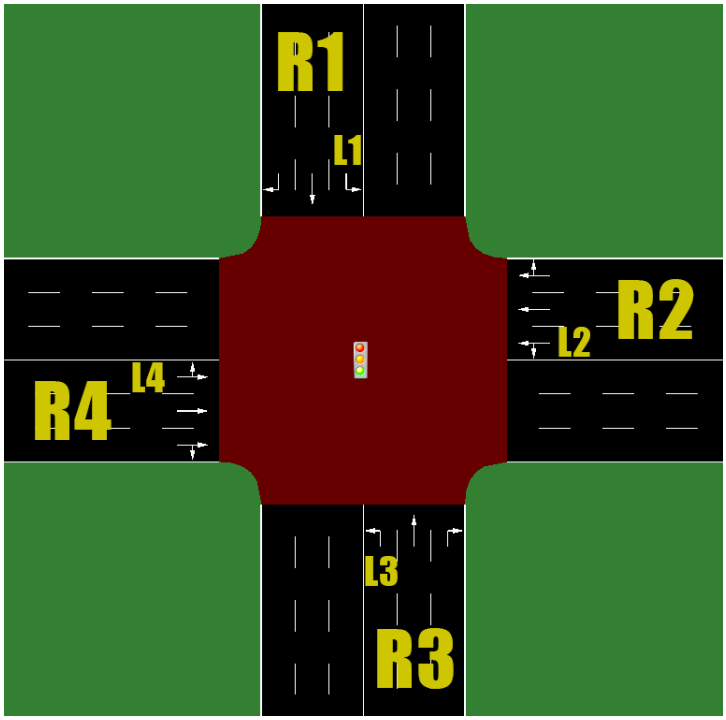


Figure C.1: Training intersection for the LuST deployment case (R=road segment, L=left turn).

Table C.1 explains the different light phases of the training and deployment intersections.

	R1	**R2**	**R3**	**R4**	**L1**	**L2**	**L3**	**L4**	**Duration**
Phase 1	R	G	R	G	R	G	R	G	Minimum: 5″ Maximum: 31″
Phase 2	R	A	R	A	R	G	R	G	4″
Phase 3	R	R	R	R	R	G	R	G	Minimum: 5″ Maximum: 6″
Phase 4	R	R	R	R	R	A	R	A	4″
Phase 5	G	R	G	R	G	R	G	R	Minimum: 5″ Maximum: 31″
Phase 6	A	R	A	R	G	R	G	R	4″
Phase 7	R	R	R	R	G	R	G	R	Minimum: 5″ Maximum: 6″
Phase 8	R	R	R	R	A	R	A	R	4″

Table C.1: Description of the 8 phases of the training and deployment intersections (R=road segment, L=left turn, colors correspond to light colors: G=green, R=red, A=amber).