A DRL solution to help reduce the cost in waiting time of securing a traffic light for cyclists.

Lucas Magnana, Hervé Rivano, Nicolas Chiabaut

A DRL solution to help reduce the cost in waiting time of securing a traffic light for cyclists.

Lucas Magnana[a], Hervé Rivano[a], Nicolas Chiabaut[b]

[a]CITI, INSA Lyon-Inria, Université de Lyon, Villeurbanne, France; [b]Département de la Haute-Savoie, Annecy, France

ARTICLE HISTORY
Compiled November 23, 2023

ABSTRACT
Cyclists prefer to use infrastructure that separates them from motorized traffic. Using a traffic light to segregate car and bike flows, with the addition of bike-specific green phases, is a lightweight and cheap solution that can be deployed dynamically to assess the opportunity of a heavier infrastructure such as a separate bike lane. To compensate for the increased waiting time induced by these new phases, we introduce in this paper a deep reinforcement learning solution that adapts the green phase cycle of a traffic light to the traffic. Vehicle counter data are used to compare the DRL approach with the actuated traffic light control algorithm over whole days. Results show that DRL achieves better minimization of vehicle waiting time at almost all hours. Our DRL approach is also robust to moderate changes in bike traffic. The code of this paper is available at `https://github.com/LucasMagnana/A-DRL-solution-to-help-reduce-the-cost-in-waiting-time-of-securing-a-traffic-light-for-cyclists.`.

KEYWORDS
Deep reinforcement learning; traffic light; cyclists; waiting time; 3DQN; actuated; bike counts; car counts;

1. Introduction

Promoting cycling as a mode of transport is prevalent in urban policies worldwide, especially to reduce CO2 emissions of transportation (Mizdrak, Blakely, Cleghorn, & Cobiac, 2019). Cycling also saves residents time, money and improves their health (Oja et al., 2011). However, the cyclability of a city depends on space intensive infrastructures, such as bike lanes separated from the flow of cars, the feeling of safety being a major criterion for cyclists (Adam, Ortar, Merchez, Laffont, & Rivano, 2022; Cervero, Caldwell, & Cuellar, 2013). The development of innovative, space-saving infrastructures is therefore becoming a necessity. In particular, traffic lights could be used to separate the flows of bikes and cars, hence securing the former. Each green phase of a traffic light allows certain vehicles that are not in conflict to pass through the intersection. Unfortunately, the green phase cycle is often independent on the traffic situation, making them unfit for bike isolation. However, recent advances in artificial intelligence for decision-making (OpenAI et al., 2019) and image recognition (Naranjo-Torres et

CONTACT Lucas Magnana. Email: lucas.magnana@gmail.com

al., 2020), among others, can enable traffic lights to adapt to the traffic (Genders & Razavi, 2019b).

1.1. Cyclists and traffic lights

Cyclists are known not to always respect red lights. The proportion of cyclists observed running a red light varies from study to study, ranging from 40% (Schleinitz, Petzoldt, Kröling, Gehlert, & Mach, 2019) to 60% (Richardson & Caulfield, 2015). Johnson, Charlton, Oxley, and Newstead (2013) showed that in Australia, where people drive on the left, cyclists are more willing to infringe a red light when they want to turn left. The authors conclude that they cannot demonstrate that running a red light increases the likelihood of an accident. The red light infringements did not result in any risk to the safety of cyclists during their observations. Hollingworth, Harper, and Hamer (2015) showed a small increase in risk of accident-related injuries for cyclists infringing red lights. They note, however, that this increase could be caused by the generally riskier behavior of cyclists running red lights rather than actually running them. Traffic light-controlled intersections are nevertheless still dangerous places for cyclists. Miranda-Moreno, Strauss, and Morency (2011) studied the cyclist injury occurrence at traffic lights, and their results suggest that cyclists safety at traffic lights is significantly affected by cyclist volumes and traffic flows. The conflicts between motorized vehicles and cyclists, especially when it comes to right-turns, seem to significantly increase the risk of collisions in Montreal, Canada, where people drive on the right. Whether in Canada or Australia, dangerous behavior performed or experienced by cyclists increases in the case of trajectories that do not involve crossing other lanes (Johnson et al., 2013; Miranda-Moreno et al., 2011). To address this issue in France, M12 signs indicate that cyclists may cross the intersection in specified directions when the light is red, with priority to vehicles with green lights.

The safety of cyclists at traffic lights is an issue. Some experiments try to help cyclists reach traffic lights when they are green. Andres, Kari, Von Kaenel, and Mueller (2019) created e-bikes designed to make cyclists catch a green wave. When the cyclist crosses the first green light, the e-bike adapts its assistance to make the cyclist go at the most adapted speed for the green wave. Similarly, Fröhlich et al. (2016) developed a smartphone application suggesting a range of speed allowing the cyclist to reach the next traffic light during a green phase. Other studies try to modify intersections for cyclists, which has the advantage of benefiting all cyclists and not just those with the right equipment. De Angelis et al. (2019) asked cyclists to rate several interfaces at traffic lights, indicating whether cyclists are on time for a green wave. Anagnostopoulos, Ferreira, Samodelkin, Ahmed, and Kostakos (2016) proposed traffic lights that prioritizes cyclists by detecting their smartphones. They however did not evaluate the impact of such a system on motorized traffic.

1.2. DRL for traffic light control

Deep reinforcement learning (DRL) has been used to adapt the behavior of traffic lights to the current traffic conditions and optimize the performance of the intersection. DRL is based on reinforcement learning (RL), an area of machine learning in which an agent develops its behavior through experience. The agent evolves in its environment and have the possibility to perform actions which modify it. At each step t, the agent receives the state of its environment $s_t \in S$ and chooses an action $a_t \in A$ with S the

set of all possible states and A the set of possible actions per state. Once the action executed, the environment sends its new state $s_{t+1} \in S$ and a reward r_t to the agent. The reward is a numerical value indicating how good or bad the action was. The goal of the agent is to develop a policy π which maps an action to a state $\pi(s) = a$ as to maximize the cumulative reward $\sum_{t=0}^{T} \gamma^t r_t$ with $\gamma \in [0, 1)$ the discount-rate weighting the distant future events. DRL algorithms are RL algorithms in which the agent uses deep learning to make decisions (further explanations are given in Section 2). Some studies use DRL to modify a traffic light's pre-defined behavior. Li Li and Wang (2016), Tan, Poddar, Sharma, and Sarkar (2019) as well as Wei, Zheng, Yao, and Li (2018) used traffic lights with a static cycle and applied DRL to optimize the changing phase timing. The agent chooses whether the light switches to the next phase or remains at the current one, with a minimum time between two phase changes. Genders and Razavi (2019a) used a traffic light with a static cycle and an initial duration for each phase. The DRL agent can increase or decrease the duration of a phase and has to find the optimum duration for each phase.

Several studies used DRL to learn a dynamic cycle. In these, the DRL agent chooses at regular intervals which phase is the best. It selects not only the timing of phase changes, but also the order in which they take place. Some authors compare their DRL approaches to a deep learning approach (Genders & Razavi, 2016) or to other DRL approaches (Mousavi, Schukat, & Howley, 2017). S. Wang, Xie, Huang, Zeng, and Cai (2019) compared their DRL approach to one static and one dynamic traffic light control method on simulations with traffic demand evolving arbitrarily. Genders and Razavi (2019a) compared their DRL approach to the same methods, but simulated peak hours to test its robustness in a more realistic setup.

1.3. Positioning and contributions

Cyclists are known to prefer infrastructure that allows them to ride away from cars (Caulfield, Brick, & McCarthy, 2012; Tilahun, Levinson, & Krizek, 2007). That's part of the reason some experimentation are planned in France to allow bikes to set off earlier in order to regain sufficient speed before the departure of other vehicles. The idea behind this work is to extend the latter concept by creating specific green phases for cyclists. This type of space-saving infrastructure would make it possible to separate bike and car flows just as well as conventional dedicated lanes, but at the expense of waiting time. Indeed, more green phases means a longer cycle, and therefore a longer waiting time between two green phases for all lanes. In this paper, we propose a DRL based green phase selection method, allowing the creation of specific green phases for cyclists with a limited impact on the waiting times at the intersection. The agent controls the order and the timing of phase changes. We use real life counts data to compare our approach to existing traffic light control methods with realistic traffic. We hope that an infrastructure of this type with a sufficiently low impact on waiting time at the intersection would foster a modal shift toward cycling, thereby further increasing cyclists' safety (Elvik, 2009). The contributions of this paper can be summarized as :

- We propose a traffic light system that is safer for cyclists as it includes specific green phases for them.
- We design a phase-change method using DRL to reduce the waiting time increase caused by such an infrastructure, using and improving existing designs.
- We use real life counts data to test our approach on a daily scale.
- We compare our approach to a dynamic one already deployed in order to demon-

strate the relevance of using a DRL based solution.

2. Deep reinforcement learning

To limit the increase in waiting time caused by the addition of green phases for cyclists, a DRL based phase-change method is proposed. This solution uses the Double Dueling Deep Q-Network (3DQN) algorithm, which is detailed in this section.

2.1. Deep Q-Network

In 2015, Mnih et al. (2015) developed an algorithm called Deep Q-Network (DQN) capable of learning human level policies. DQN is based on a reinforcement learning algorithm called Q-learning (Watkins & Dayan, 1992). A function $Q : S \times A \rightarrow \mathbb{R}$ calculates the quality of a state–action combination. Every time the agent chooses an action a_t, $Q(s_t, a_t)$ is updated using the Bellman equation. The final policy chooses the action with the best Q-value $\pi(s) = \max_a Q(s, a)$. In DQN, a deep neural network called Q-network approximates Q and is noted $Q(s, a; \theta)$ where θ represents the parameters (i.e. the weights) of the neural network. The Q-network is trained by minimizing the loss function L defined as :

$$L(\theta) = (Y_t^{DQN} - Q(s, a; \theta))^2$$

$$Y_t^{DQN} = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta'))$$

Y_t^{DQN} is called the target value and θ' represents the parameters of the target network, a second neural network with the same architecture as the Q-network. The target network is only used to compute Y_t and is updated towards the Q-network during training.

2.2. Double Deep Q-Network

In 2016, van Hasselt, Guez, and Silver (2015) have shown that DQN has an overestimation bias for Q-values and proposed a solution inspired by the double Q-learning algorithm (Hasselt, 2010) called Double Deep Q-network (DDQN). DDQN is very similar to DQN but calculates the target value a little differently:

$$Y_t^{DDQN} = r_t + \gamma Q(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta); \theta')$$

In DQN, the action a' used to calculate the target value is chosen and evaluated by the target network where in DDQN, the action is chosen by the Q-network and evaluated by the target network. This reduces the overestimation of Q-values, thus increasing the quality of the policies produced.

2.3. Dueling Deep Q-Network

The advantage function is defined as $A(s,a) = Q(s,a) - V(s)$ where $V(s)$ represents the expected long term reward for being in the state s. From this equation, Q can be decomposed as the sum of $V(s)$ and $A(s,a)$. The idea behind Dueling Deep Q-Network developed by Z. Wang et al. (2016) is to decompose the Q-network in two streams : $V(s;\theta)$ which approximates $V(s)$ and $A(s,a;\theta)$ which approximates $A(s,a)$. The Q-network and the target network are therefore defined as :

$$Q(s,a;\theta) = V(s;\theta) + (A(s,a;\theta) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta))$$

$$Q(s,a;\theta') = V(s;\theta') + (A(s,a;\theta') - \frac{1}{|A|}\sum_{a'} A(s,a';\theta'))$$

with θ the parameters of the Q-network and θ' the parameters of the target network. The mean of the approximations of advantages is subtracted from the approximations of $A(s,a)$ to increase learning stability and performance.

2.4. Double Dueling Deep Q-Network (3DQN)

The Double DQN and the Dueling DQN can be combined to obtain the Double Dueling Deep Q-Network (3DQN). 3DQN shows better learning stability and performance than DQN or DDQN in general. In 3DQN and all algorithms derived from DQN, the agent doesn't learn after every action it performs. Instead, it has a memory buffer in which it stores all the transitions $(s_t, a_t, s_{t+1}, r_t, d)$. This transition means that during the step t, the agent received s_t, chose the action a_t which was rewarded by r_t and made the environment in the state s_{t+1}. d contains the information whether s_{t+1} is a final state or not. The memory buffer has a finite size and if a new transition needs to be stored once it's full, the oldest is replaced by the new one. At each learning phase, the agent randomly fills a batch with transitions contained in the memory buffer and computes their mean loss. The mean loss is backpropagated to modify the weights of the Q-network. In our implementation, the weights of the target network are periodically replaced by the weights of the Q-network. Finally, a ϵ -greedy policy is used during training. When the agent needs to choose an action, it computes the Q-values using the Q-network. A random number $r \in [0,1]$ is generated. If $r < \epsilon$, the action is chosen randomly. Otherwise, the action with the highest Q-value is chosen. ϵ is set to 1 at the beginning of the training and decreases as training progresses, in order to have a lot of exploration at the beginning and a lot of exploitation at the end.

3. 3DQN approach

This section presents all the components the 3DQN agent will need to approximate an optimal policy, as well as how it is trained. First, the type of environment in which the agent will evolve is detailed. Then, the type of states that the environment will send to the agent is explained. The actions the agent will be able to perform in the

environment, as well as the function rewarding them are defined. Finally, the training process and all the implementation choices made are explained.

3.1. Environment

As explained in Section 1.3, the traffic light has a green phase for each incoming car lanes but also for each incoming bike lanes. The environment in which the agent will evolve in is thus an intersection, made up of several intersecting axes, and controlled by a traffic light. For the sake of simplicity, each axis is assumed to be two-way, with a bike lane in each direction. If the car lanes on a given axis all have a green light at the same time (i.e. there are no specific green phases for turning cars), the light at an intersection of n axes will have $2n$ different green phases. For example, the set of green lanes on an intersection of 2 axes will be $G = \{g_{car}^{ax_1}, g_{bike}^{ax_1}, g_{car}^{ax_2}, g_{bike}^{ax_2}\}$ with $g_t^{ax_x}$ meaning that the vehicle type t on the axis x has the green light. A graphical example of a two axes intersection is shown in Figure 1, with ax_1 being the North-South (N-S) axis and ax_2 the East-West (E-W) axis.

3.2. States

The states sent to the agent need to condense the useful information about the environment. We identified two type of information about the vehicles at the intersection that the agent needs to be informed of. First, the position of the vehicles. To make the best decision, the agent needs to know how many vehicles arrive at the intersection and on which lane they are. The second one is the speed. The agent needs to make the difference between the vehicles that are waiting and the one that are moving. The more vehicles waiting in a lane, the more it is necessary to change phase to let them pass. As in the work of Liang, Du, Wang, and Han (2019), the states are two matrixes of same dimensions. They are named respectively the position matrix and the speed matrix. The environment is divided in squares of 5-meters long. Only the squares belonging to the lanes arriving at the intersection are put in these matrices. The other ones can not contain useful information. The matrixes are thus smaller than those of Liang et al. (2019), as their dimensions are $N \times P$ with N the number of lanes arriving at the intersection and P the number of squares each lane contains. The position matrix contains the number of vehicles that are in each square, and the speed matrix contains the mean speed of the vehicles in each square. These matrixes could be reconstructed with cameras pointed at the lanes arriving at the intersection, and recent methods for estimating vehicle position and speed Gunawan, Tanjung, and Gunawan (2019). A graphical example of a position matrix is shown in Figure 1.

3.3. Actions

The actions performed by the agent need to modify the behavior of the traffic light at an intersection. As it has been done several times (Genders & Razavi, 2019a; Mousavi et al., 2017; S. Wang et al., 2019), the set of green phases G is used as the action space (see Section 3.1). Once a green phase start, 10s pass before the agent chooses the future green phase. If the chosen green phase is different from the actual one, a 4s orange light phase is triggered for the lanes that had green light until the decision. After this orange phase, or if the chosen green phase is the same as the actual one, a new 10s period is started before the agent chooses again. Waiting 10s after the start

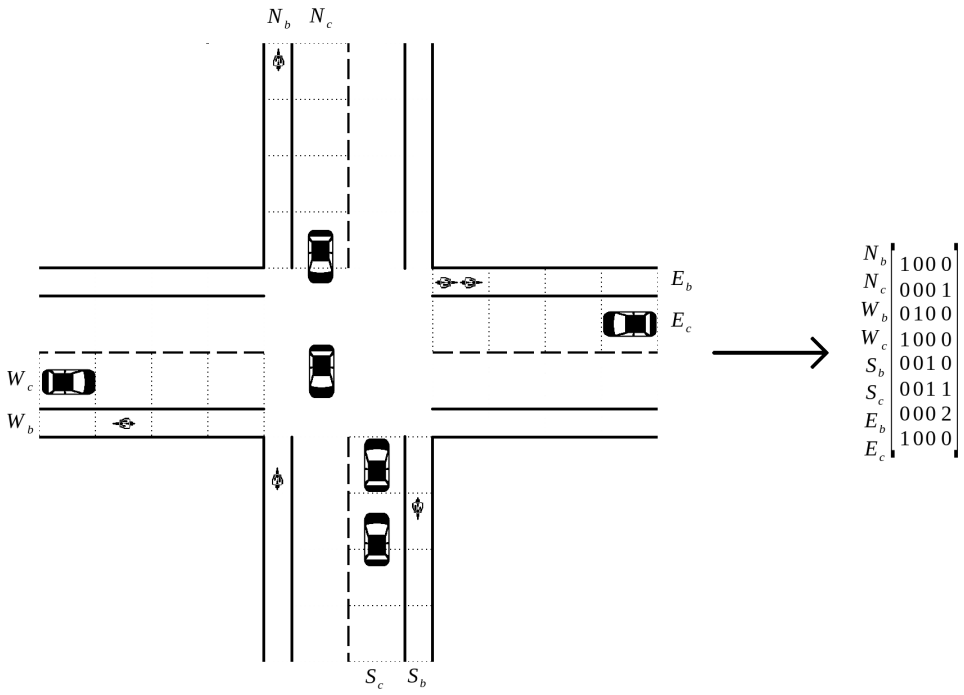


Figure 1. Diagram showing the construction of the position matrix from an image of an intersection.

of a green phase avoids sudden changes that could surprise vehicles, and increases the stability of the agent's learning.

3.4. Rewards

Genders and Razavi (2019a) used the same action space and developed a reward function working with it. The rewards they used are adapted as explained below.

3.4.1. Reward function

The reward function is defined as :

$$r_t = -(w_b + w_c)^2$$

with w_b and w_c being respectively the number of waiting bikes and the number of waiting cars. A vehicle is considered to be waiting when its speed is less than 0.5 km/h. Note that the reward can only be negative, and that the more vehicles waiting, the more negative the reward. The agent must minimize the number of waiting vehicles in order to maximize the reward. The sum of w_b and w_c is squared to to discriminate more strongly against bad decisions and facilitate the start of the training.

3.4.2. Scaling factor

However, the large negative values that the reward function can take, especially at the start of training, may hamper the agent's convergence. Genders and Razavi (2019a)

coped with this issue by dividing the rewards by r_{max}, the biggest reward calculated. In our case, this normalization allowed the agent to converge, but has not led to the creation of effective policies. r_{mean}, the mean of all calculated rewards, is therefore used instead. All the calculated rewards as well as the number of actions performed by the agent during training are stored, and r_{mean} is updated at the end of each training episodes (which are detailed in Section 3.5). Using r_{mean} as a scaling factor incites the agent to perform actions that are better on average than those it has performed so far.As the agent improves, the average rewards will increase, pushing the agent ever further to become better. This allows the agent to finish training with a high-performance policy.

3.5. Training

3.5.1. Q-network architecture

A state is made up of two matrixes that can be seen as a two channels image condensing the useful information of traffic at the intersection. Thus, the Q-network needs to be able to find patterns in an image in order to correctly process the states it receives. Convolutional layers extract features from images to lower dimensions without loosing their characteristics. A convolutional layer is composed of kernels, which are matrixes of small dimensions. Each kernel has its own weights in order to find a specific type of pattern in the image. The kernels slide along the image, and a multiplication is performed between their weights and the image values for each sub-area they cover. The Q-network is composed of two convolutional layers containing 16 kernels of dimension 2x2. These layers are followed by two fully connected layers of 128 nodes each. Then comes two output layers for the value function and the advantage function (see Section 2.3).The ReLU activation function is used between all layers in order to provide non-linear properties to the Q-network. Figure 2 summarizes the architecture of the Q-network.

3.5.2. Hyperparameters

The values of the hyperparameters used during training as well as all the variables used in this paper are shown in Table A1 in Appendix A. The agent is driven in episodes. During each episode, vehicles can spawn during 6 simulated hours, one step per second.An episode ends when all vehicles have spawned and no vehicle remain in the simulation. A vehicle disappears once it has reached its destination, which is the end of one of the lanes leaving the intersection. After acting pt times, the agent learns at the end of each episode. Training stops when the agent has made f actions. ϵ decreases linearly each time the agent chooses an action, and reaches its ending value at the f^{th} action. Finally, the target network is updated by replacing its weights with those of the Q-network every υ actions performed by the agent.

4. Experimental setup

In this section, the simulated environment(cyclists, cars, and traffic lights) is presented. Then the synthesis of the trafficbased on real count data is detailed. Finally, our performance evaluation methodology is explained.

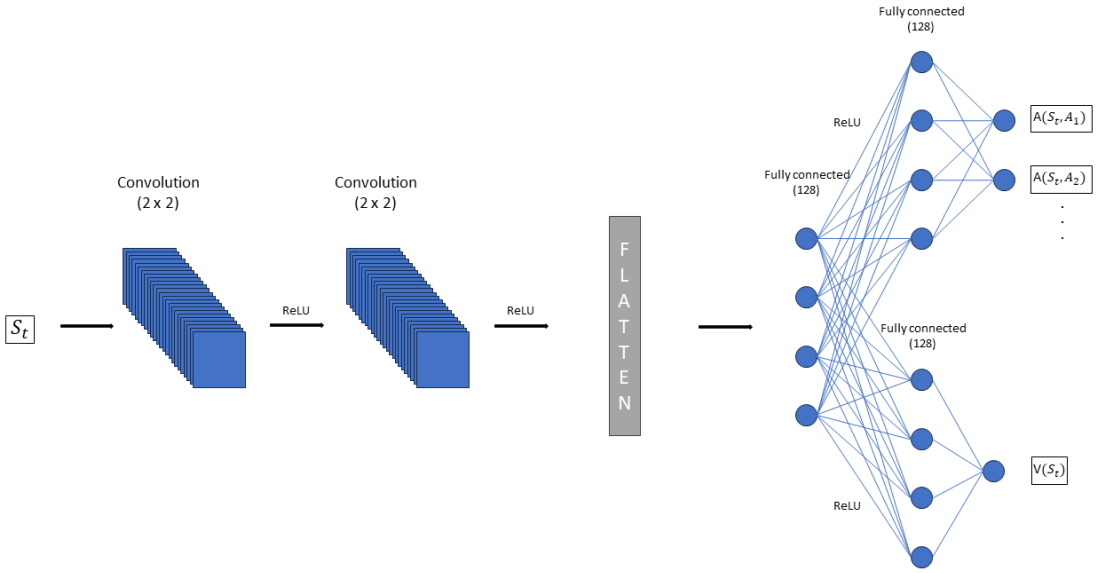


Figure 2. Diagram showing the structure of the Q-network.

4.1. Simulated environmnet

SUMO (Simulation of Urban MObility) is used to simulate the environment. SUMO is a tool that allows different actors to interact on a road graph. The behavior of the different actors can be changed in real time. SUMO is commonly used in traffic light control studies using simulation. Figure 3 shows a screenshot of the environment. An intersection with two axes crossing (NS for North-South and EW for East-West) is managed by a traffic light. Each road arriving at the intersection is 150m long and has two lanes, one for bikes and one for cars. The traffic light has four green phases $G = \{g_{car}^{NS}, g_{bike}^{NS}, g_{car}^{EW}, g_{bike}^{EW}\}$. g_{bike}^{NS} is activated on Figure 3. The vehicles spawn at the edge of a road and have the edge of another road as their destinations. When a vehicle is on green and wants to turn left, it must give way to oncoming traffic before crossing the intersection. This adds a waiting time that does not depend on the green phase of the traffic light. Moreover, when possible, vehicles tend to position themselves in the middle of the intersection when waiting to turn left to let vehicles behind them pass. Unfortunately, SUMO doesn't allow this behavior, making everyone wait behind it when a vehicle wanting to turn left is waiting to do so. This adds even more waiting time not dependent on the state of the traffic light. Vehicles are therefore prohibited from turning left. Vehicles have equal probabilities of having as destination one of the two roads they can access without turning left.

4.2. Vehicle counts and traffic synthesis

The city of Paris makes automatic vehicle counter data available on its open-data website [1]. Various temporal aggregations are available, from the year to the hour.

[1] https://opendata.paris.fr/pages/home/

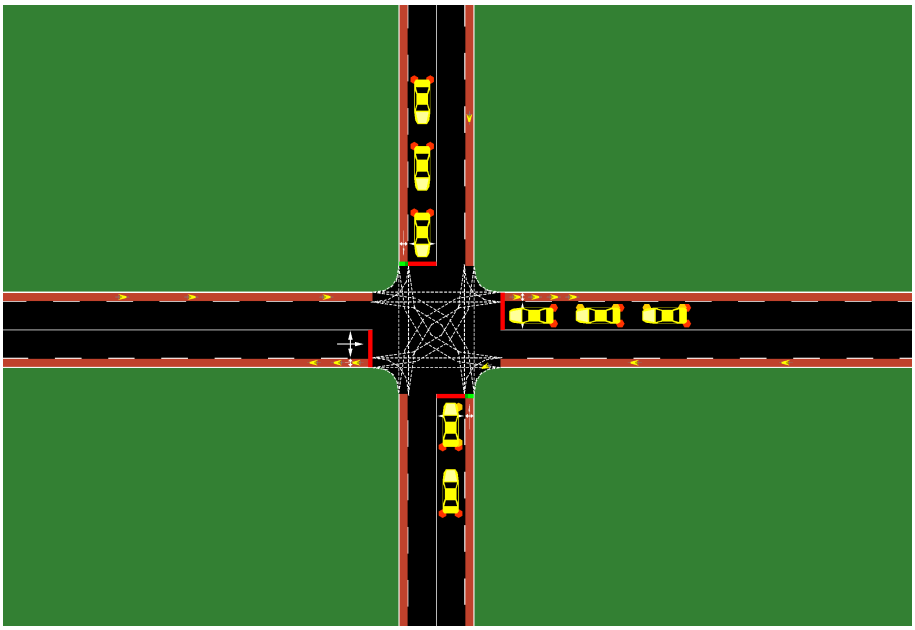


Figure 3. Screenshot of the environment simulated by SUMO.

Hourly aggregation is used here as it is the most precise. The data of two unidirectional car counters and one bidirectional bike counter are collected. The counters are located on boulevard Montparnasse and are close to each other. The boulevard Montparnasse is two-way, with two car lanes and a bike lane in each direction. The number of counted cars is halved, since our simulation has a single car lane in each direction. The data are from June 20, 2023, a Tuesday with good weather.

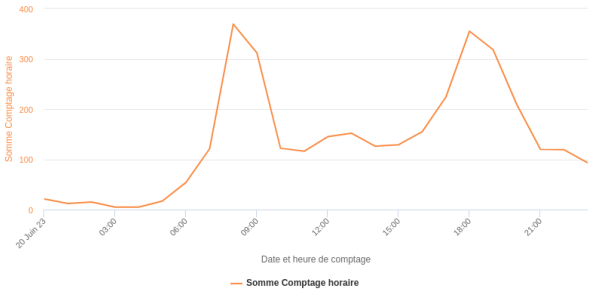
Figure 4 shows the sum of the number of vehicles counted in both directions per hour. Differences between car and bike distribution are observable. There are hardly any bikes counted at night, a relatively stable number of bikes from 10:00 to 16:00 and two huge peaks at 08:00 and 18:00, the usual commuting hours. For the cars, the number decreases during all the night before increasing again at 06:00. The counted cars then reach a plateau that lasts until 17:00. Then comes a small increase with a peak at 19:00 before a decrease that will last until late at night. These differences between the vehicles distribution shows the importance of using real data on a daily scale. The 3DQN approach must be able to adapt its decisions to changes in both car and bike traffic in order to be efficient on a daily scale.

In order to simulate the traffic at the scale of each vehicle, we assume that the number of vehicle arriving at lane l each second follows a Poisson process p^l. The intensity $\lambda_{p^l}(t)$ of this process at time t is considered fixed during each hour and follows the aggregated count data :

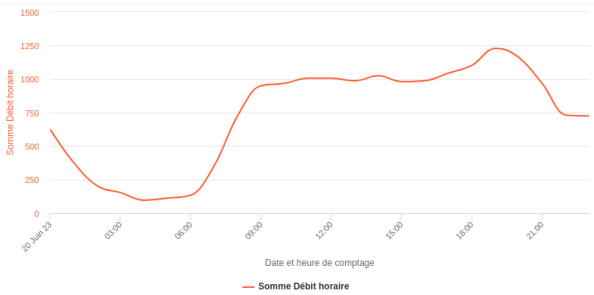
$$\lambda_{p^l}(t) = \frac{c^l_{h(t)}}{3600}$$

with $c^l_{h(t)}$ the number of vehicles (bikes or cars) counted at hour $h(t)$ on lane l.

To summarize, the environment simulates the crossing of two Montparnasse boulevards during one day, each with only one car lane, and with traffic synthesized by



(a) Hourly bike count.



(b) Hourly car count.

Figure 4. Average number of vehicles counted per hour in both directions on boulevard Montparnasse on June 20, 2023.

extrapolating the hourly aggregation of real vehicles counts.

4.3. Performance evaluation methodology

In our settings, the traffic is never saturated at the exit of the intersection. The performance of a solution is therefore the time spend by vehicles before they reach the intersection and get a green light. After training, whole days ($3600 \times 24 = 86400$ steps) are simulated and the mean waiting time of vehicles cars is calculatedfor each hour. The 3DQN approach is compared to the following traffic light control mechanisms.

- *static unsecured* : The first approach compared, named *unsecured* in the following, serves as a baseline to quantify the addition of waiting times when securing the traffic light for cyclists. This is a classic static traffic light, with only one green phase per axis. The bikes and the cars on the same axis cross the intersection at the same time.
- *static secured* : This approach is a naive one. It simply consists in preventing bikes from passing during the existing green phases and adding a green phase for bikes to all axes containing at least one bike lane. In our case, the traffic light has four green phases when using this approach, each lasting 40s. This behavior shows the huge increase in waiting time for all vehicles if the bike safety system is naively implemented.
- *actuated* : The *static secured* approach serves mainly to demonstrate the importance of a dynamic phase-change method when implementing specific green phases for bikes. Comparing the 3DQN approach which is highly-dynamic only to a static approach would not be fair. Thus, *actuated* is used. *actuated* is a dynamic

phase-change method commonly implemented in Germany (Brilon & Laubert, 1994). A traffic light in *actuated* mode has vehicle detectors on each of its incoming lane, approximately 50m ahead. The traffic light has a *duration* parameter, and each green phase has a minimum duration $minDur$ and a maximum duration $maxDur$. When a green phase start, the traffic light waits $minDur$ seconds before starting a counter of *duration* seconds. Once the counter reaches zero, the traffic light switches to the next phase of its cycle. If one of the detector on the lanes with the green light detects a vehicle before the counter reaches zero, the counter is reset. If a green phase reaches a duration of $maxDur$, the traffic light switches to the next phase, regardless of the counter's status. In the implementation of *actuated* used, all green phases have a $minDur$ of 10s, a $maxDur$ of 40s and the *duration* parameter is set to 5s. *actuated* is commonly used to test the performance of DRL approaches doing traffic light control (Genders & Razavi, 2019a; Tan et al., 2019; S. Wang et al., 2019).

5. Results

The results are in two parts. The first ones are on an hourly scale for a simulation of one day. An initial simulation with a traffic light controlled by the 3DQN agent is carried out, with our random traffic synthesis. The times and places vehicles appear during this simulation are recorded. Three further simulations, one for each other control mechanisms, are then carried out with the traffic trace of the first simulation. This allows a comparison under perfectly identical conditions. The second part of the results is at a larger scale. The traffic demand of bikes is changed, and five different initial simulations are made for each bike traffic demand. The *actuated* approach is then used for each initial simulation in the same way as explained above.

5.1. Hourly results

Figure 5 shows the results produced by a simulation of one day. The x-axis shows the hours of the day, from 0 to 23, and the y-axis shows the number of vehicles on Figure 5a and the mean waiting time of the vehicles on Figure 5b. The vehicle distribution curves have the same shape as those shown in Figure 4 without being perfectly identical. This is due to the randomness of the Poisson processes.

As expected, the *unsecured* approach does better than all the other ones. Unlike other curves, the *unsecured*'s one is flat and changes little with traffic. The traffic is indeed not strong enough to saturate the road graph in this configuration. All vehicles waiting at a red phase are able to cross the intersection on the first green phase they are granted. Adding green phases dedicated to cyclists increases the occupancy rate of lanes since car lanes and bike lanes on the same axis are then emptied successively.

As expected, the worst approach is the *static secured* one. Doubling the duration of the traffic light's cycle results in an explosion of the waiting time, as each lane has to wait longer between two green phases. Reducing green phase durations to 20s increases the average waiting time even further, with lanes becoming more saturated because they don't have enough time to empty during their green phases.

actuated does much better than the *static secured* approach whatever the time of the day, and handles the double car-bike peak at 19:00 with much greater ease. As the green phases duration are adapted to the traffic situation, the lanes are much less saturated. The green light time wasted for empty lanes is greatly reduced.

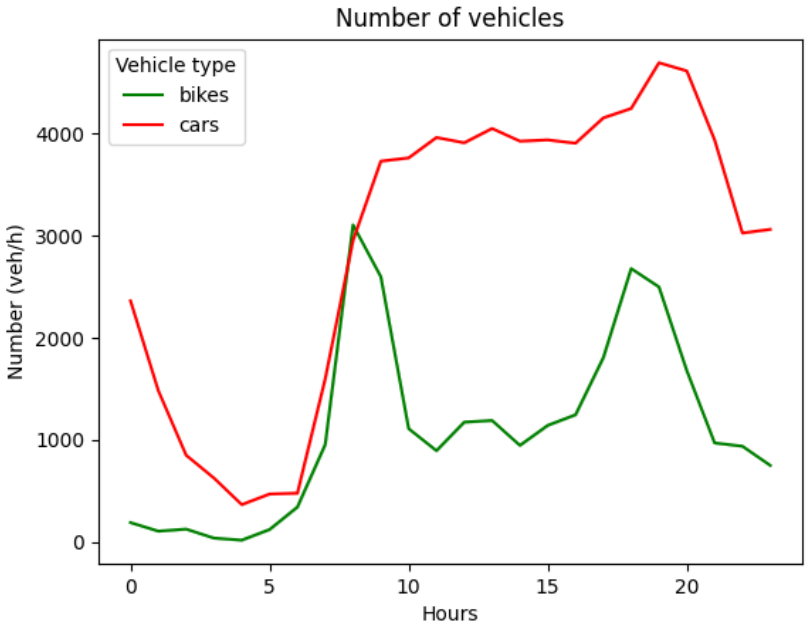
The 3DQN approach does even better than *actuated*, with a lower mean waiting time at almost every hour. The gain in waiting time is significant during low-traffic hours. This is logical, as *actuated* is less accurate when lanes are empty or almost empty. Indeed, an *actuated* traffic light must wait a minimum of 15s for each green phase and follows a static cycle, regardless of traffic conditions. 3DQN is able to detect vehicles and change the green phase accordingly. But as lanes fill up, the cycle becomes less important than the phase change timing. The 3DQN appoach being able to do both, it is still slightly better than *actuated* during high-traffic hours. The performance of the 3DQN approach is closest to that of the *unsecured* one. It's worth noting, however, that the less traffic there is, the better the 3DQN approach performs.

On the scale of the day, adding specific green phases for cyclists multiplies the mean waiting time by 4.35 when using the *40s* naive approach, by 1.69 when using the *actuated* approach and by 1.55 when using the 3DQN approach. Working with higher traffic would certainly allow us to reach the saturation rate of the *unsecured* method, where it would be less effective, but this saturation would be all the more noticeable with the addition of green lanes for cyclists. This would further degrade the performance of the other approaches, and possibly even prevent the agent's learning process.

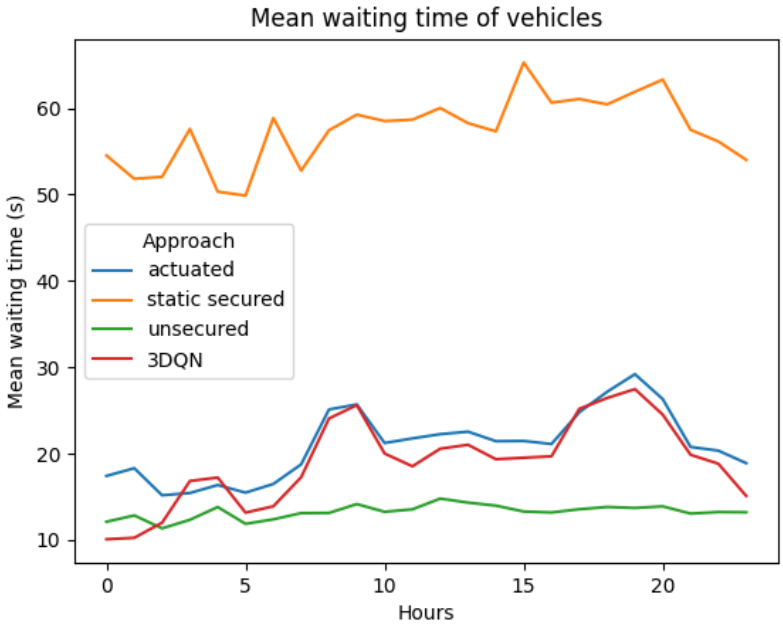
5.2. Robustness to changes in bike traffic

The 3DQN approach limits the increase of the mean vehicle waiting times on a day with traffic similar to that used during training. However, the choice of whether to cycle is strongly correlated with the weather. As the traffic demands are calculated on the basis of data from a sunny day, the number of cyclists is likely to be lower on bad weather days. On the other hand, the aim of the secured intersection is to provide safe passage for cyclists, and the deployment of such infrastructure could attract cyclists, leading to an increase in bike traffic. The robustness of a 3DQN approach to changes in bike traffic therefore appears to be an important point to check. A multiplying coefficient which goes from 0.5 to 1.5 in steps of 0.1 is set. The number of bikes counted each hour is multiplied by this coefficient. That varies the bike traffic linearly from 50% to 150% of the observed one in the count data. Five days are simulated for each new spawn rate. *actuated* being the best comparison approach on a secured traffic light, it is used to evaluate the performance of 3DQN. Since night-time hours are not very relevant due to the absence of traffic, the results shown in Figure 6 and 7 focus on the hours between 6h and 20h.

Figure 6 shows the number of vehicles (6a) and the sum of all the waiting times (6b) for each multiplying coefficient. Logically, the number of cars per simulation is stable and the number of bikes is increasing linearly. The sum of vehicle waiting times generated by 3DQN starts out lower than the one generated by *actuated*. The two increase progressively, finally coming together when the coefficient reaches 1.5. When the coefficient is 1.5, 3DQN makes vehicles wait longer than *actuated*. This is consistent with hour-by-hour observations. The fewer vehicles there are at the intersection, the higher the probability that *actuated* will leave an empty lane green, due to its fixed cycle and minimum green phase time. This logically favors 3DQN, which is by nature more dynamic. It still shows that 3DQN is able to adapt to a decrease in bike traffic without any significant impact on its decision-making performance. This is probably due to the high variations in both car and bike traffic that the agent faces during training, with nighttime hours having very low traffic levels. 3DQN's performance is



(a) Hourly number of vehicles.



(b) Hourly mean waiting time.

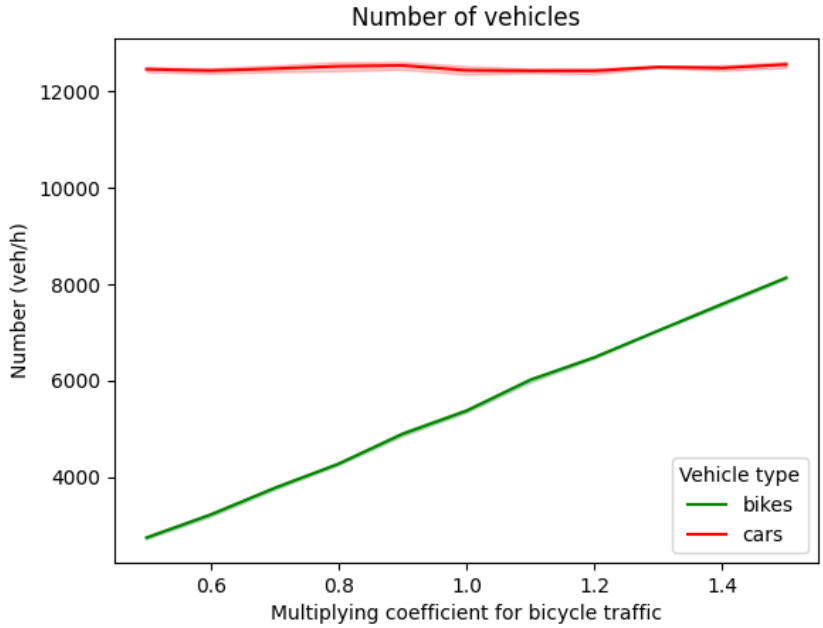
Figure 5. Hourly number of vehicles and mean waiting time for a simulation of one day.

also fairly stable as the number of bikes increases. The sum of waiting times logically increases, as more bikes means more green light time is needed to clear the bike lanes, which impacts the waiting times of all vehicles at the intersection. However, the difference in performance between the two approaches diminishes as the coefficient increases, until it reaches 1.5. When the multiplying coefficient reaches 1.5, not only does 3DQN make vehicles wait longer on average than *actuated*, but the discrepancy in performance between simulations also increases significantly. The first reason for this is that, as explained above, the more vehicles there are, the better the *actuated* performance. But it's also probably due to less relevant decisions made by 3DQN. The further the bike traffic moves away from the one used during training, the greater the probability that the agent will receive a state it is not accustomed to handling. This situation may lead the agent not to make the best possible decision, resulting in more saturated lanes that it is no longer able to manage properly. The 3DQN agent therefore appears to be robust with bike traffic ranging from 50% to 140% of the traffic used during training.

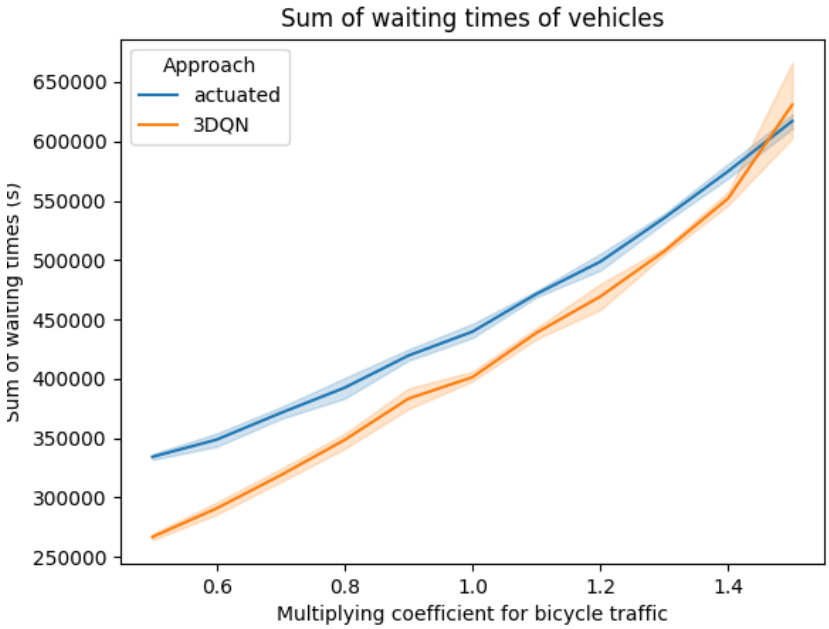
To go into more details, Figure 7 shows the average waiting time for bikes (7a) and cars (7b). For *actuated*, mean waiting times evolve in a fairly similar way for both types of vehicle, with a slight linear increase. 3DQN is different. For cars, there is also a linear increase, but much more pronounced to the point where the average waiting time for cars observed with 3DQN exceeds that of *actuated* when the multiplier coefficient reaches 1.1. On the other hand, the average waiting time for bikes is stable and even decreases until the coefficient reaches 1.1, before rising slightly. This is a surprise, as we expected the mean waiting times for the two types of vehicles to evolve in the same way as the sum of the waiting times in Figure 6. Instead, the average waiting time for cars increases more sharply, allowing bikes to wait less. The agent seems to wait for a lane to reach a certain occupancy rate before turning it green, thus favoring cyclists in the experiment. Indeed, as the number of cyclists spawning in the simulations increases, the bike lanes reach this occupancy rate more quickly, prompting the agent to turn them green more often. As a result, the waiting time for bikes does not increase despite their higher numbers, but cars are given the green light less often. That's why the mean waiting time for cars increases this way. The sum of mean waiting times observed with 3DQN ends up exceeding that of *actuated*, because by giving preference to bikes in this way, the mean waiting time for cars ends up being too great for the agent to be as efficient as *actuated*.

6. Conclusion

In this paper, we proposed a traffic light allowing cyclists to cross an intersection safely during dedicated green phases. Results show that adapting the behavior of a traffic light to the traffic situation using DRL can reduce the cost in waiting time induced by securing the passage of cyclists with dedicated green phases. A 3DQN agent is capable of controlling this type of secure traffic light with different levels of traffic, enabling it to absorb fluctuations in traffic on a typical day. Performance remains relatively stable with moderate deviations from training conditions in terms of bike traffic. However, even though real count data are used to simulate traffic, it is modelled with Poisson processes, which is an unrealistic simplification of traffic demand. In the same spirit of realism, the vehicles in the simulations use SUMO's default behavior, which perfectly respects the rules of the road. Experiments with traffic demand and individual behaviors closer to reality must be carried out before such an infrastructure

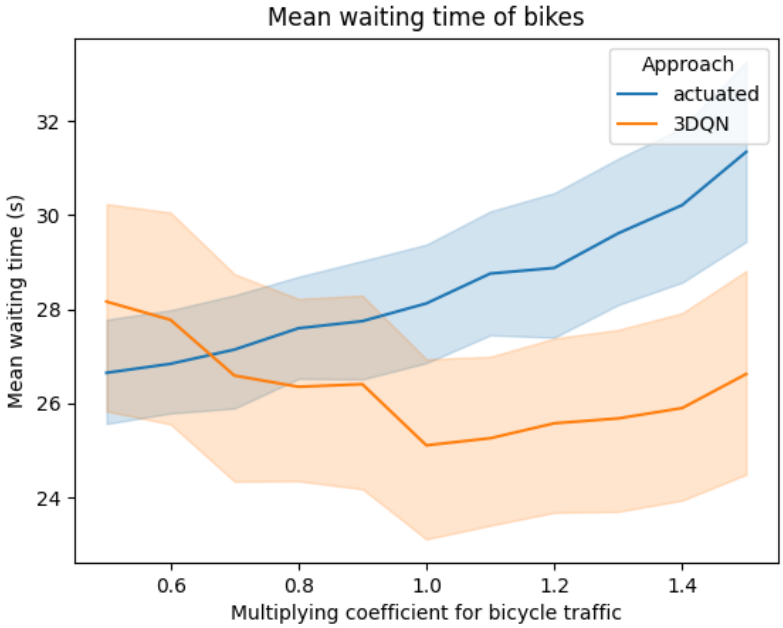


(a) Number of vehicles.

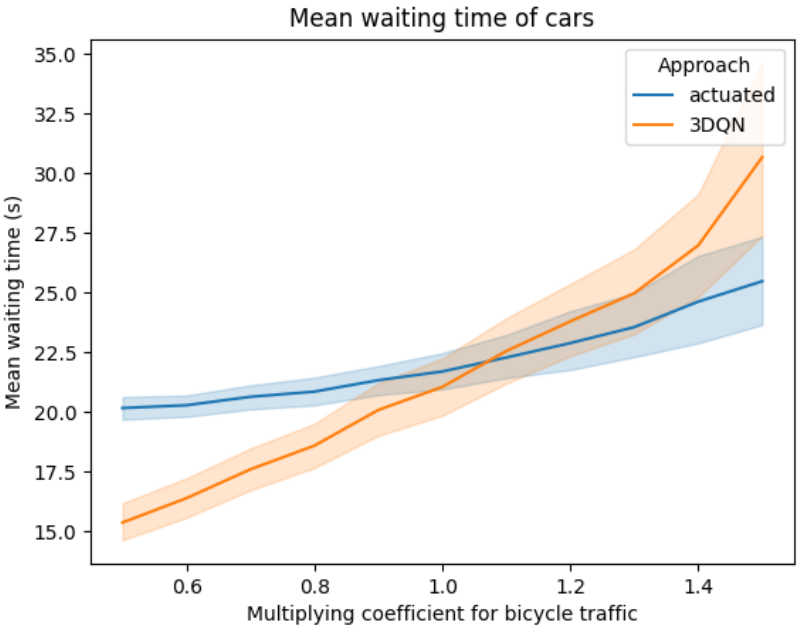


(b) Sum of waiting times.

Figure 6. Number of vehicles and sum of waiting times with respect to bike traffic changes (from 6h to 20h).



(a) Mean waiting times of bikes.



(b) Mean waiting times of cars.

Figure 7. Mean waiting times of vehicles with respect to bike traffic changes (from 6h to 20h).

can be deployed. If these experiments prove conclusive, an interesting extension of our work would be to observe the distribution of vehicles exiting the intersection and to train another DRL agent with these distributions, with the aim of creating DRL driven green waves along a path with several intersections. We would also like to point out that an agent has been trained using another type of (policy-based) DRL algorithm called Proximal Policy Optimization (PPO). Although this agent has converged, its final policy performs less well than that of the 3DQN agent, but we don't know whether this is due to the nature of PPO or to our implementation. We are making available the code containing both algorithms for future works.

Disclosure statement

The authors declare that they have no relevant or material financial interests that relate to the research described in this paper.

References

Adam, M., Ortar, N., Merchez, L., Laffont, G.-H., & Rivano, H. (2022, January). Conducting Interviews with Maps and Videos to Capture Cyclists' Skills and Expertise. In U. of Chester Press (Ed.), *Becoming Urban Cyclists: From Socialization to Skills* (p. 18-43). University of Chester Press. Retrieved from https://hal.science/hal-03552634

Anagnostopoulos, T., Ferreira, D., Samodelkin, A., Ahmed, M., & Kostakos, V. (2016). Cyclist-aware traffic lights through distributed smartphone sensing. *Pervasive and Mobile Computing*, *31*, 22–36. Retrieved 2022-10-28, from https://linkinghub.elsevier.com/retrieve/pii/S1574119216000249

Andres, J., Kari, T., Von Kaenel, J., & Mueller, F. F. (2019). "co-riding with my eBike to get green lights". In *Proceedings of the 2019 on designing interactive systems conference* (pp. 1251–1263). ACM. Retrieved 2023-07-24, from https://dl.acm.org/doi/10.1145/3322276.3322307

Brilon, W., & Laubert, W. (1994). Priority for public transit in germany. *Journal of Advanced Transportation*, *28*(3), 313–340. Retrieved 2023-07-27, from https://onlinelibrary.wiley.com/doi/10.1002/atr.5670280309

Caulfield, B., Brick, E., & McCarthy, O. T. (2012). Determining bicycle infrastructure preferences – a case study of dublin. *Transportation Research Part D: Transport and Environment*, *17*(5), 413–417. Retrieved 2023-07-24, from https://linkinghub.elsevier.com/retrieve/pii/S1361920912000363

Cervero, R., Caldwell, B., & Cuellar, J. (2013). Bike-and-ride: Build it and they will come. *Journal of Public Transportation*, *16*(4), 83–105. Retrieved 2023-07-26, from http://scholarcommons.usf.edu/jpt/vol16/iss4/5/

De Angelis, M., Stuiver, A., Fraboni, F., Prati, G., Puchades, V. M., Fassina, F., ... Pietrantoni, L. (2019). Green wave for cyclists: Users' perception and preferences. *Applied Ergonomics*, *76*, 113–121. Retrieved 2023-07-24, from https://linkinghub.elsevier.com/retrieve/pii/S0003687018307385

Elvik, R. (2009). The non-linearity of risk and the promotion of environmentally sustainable transport. *Accident Analysis & Prevention*, *41*(4), 849–855. Retrieved 2023-07-25, from https://linkinghub.elsevier.com/retrieve/pii/S0001457509000876

Fröhlich, S., Springer, T., Dinter, S., Pape, S., Schill, A., & Krimmling, J. (2016). BikeNow: a pervasive application for crowdsourcing bicycle traffic data. In *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing: Adjunct* (pp. 1408–1417). ACM. Retrieved 2022-10-28, from https://dl.acm.org/doi/10.1145/2968219.2968419

Genders, W., & Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. *arXiv.org perpetual, non-exclusive license*.

Genders, W., & Razavi, S. (2019a). Asynchronous n -step q-learning adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, *23*(4), 319–331. Retrieved 2023-02-02, from https://www.tandfonline.com/doi/full/10.1080/15472450.2018.1491003

Genders, W., & Razavi, S. (2019b). An open-source framework for adaptive traffic signal control. *arXiv.org perpetual, non-exclusive license*.

Gunawan, A. A. S., Tanjung, D. A., & Gunawan, F. E. (2019). Detection of vehicle position and speed using camera calibration and image projection methods. *Procedia Computer Science*, *157*, 255–265. Retrieved 2023-07-20, from https://linkinghub.elsevier.com/retrieve/pii/S187705091931083X

Hasselt, H. (2010). Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), (Vol. 23). Curran Associates, Inc.

Hollingworth, M. A., Harper, A. J., & Hamer, M. (2015). Risk factors for cycling accident related injury: The UK cycling for health survey. *Journal of Transport & Health*, *2*(2), 189–194. Retrieved 2023-07-25, from https://linkinghub.elsevier.com/retrieve/pii/S221414051500002X

Johnson, M., Charlton, J., Oxley, J., & Newstead, S. (2013). Why do cyclists infringe at red lights? an investigation of australian cyclists' reasons for red light infringement. *Accident Analysis & Prevention*, *50*, 840–847. Retrieved 2023-07-24, from https://linkinghub.elsevier.com/retrieve/pii/S000145751200262X

Liang, X., Du, X., Wang, G., & Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology*, *68*(2), 1243–1253. Retrieved 2023-01-06, from https://ieeexplore.ieee.org/document/8600382/

Li Li, Y. L., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, *3*(3), 247–254. Retrieved 2023-02-02, from http://ieeexplore.ieee.org/document/7508798/

Miranda-Moreno, L. F., Strauss, J., & Morency, P. (2011). Disaggregate exposure measures and injury frequency models of cyclist safety at signalized intersections. *Transportation Research Record: Journal of the Transportation Research Board*, *2236*(1), 74–82. Retrieved 2023-07-24, from http://journals.sagepub.com/doi/10.3141/2236-09

Mizdrak, A., Blakely, T., Cleghorn, C. L., & Cobiac, L. J. (2019). Potential of active transport to improve health, reduce healthcare costs, and reduce greenhouse gas emissions: A modelling study. *PLOS ONE*, *14*(7), e0219316. Retrieved 2023-07-26, from https://dx.plos.org/10.1371/journal.pone.0219316

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. Retrieved 2023-07-07, from https://www.nature.com/articles/nature14236

Mousavi, S. S., Schukat, M., & Howley, E. (2017). Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, *11*(7), 417–423. Retrieved 2023-01-06, from https://onlinelibrary.wiley.com/doi/10.1049/iet-its.2017.0153

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. *Applied Sciences*, *10*(10), 3443.

Oja, P., Titze, S., Bauman, A., De Geus, B., Krenn, P., Reger-Nash, B., & Kohlberger, T. (2011). Health benefits of cycling: a systematic review: Cycling and health. *Scandinavian Journal of Medicine & Science in Sports*, *21*(4), 496–509. Retrieved 2023-07-26, from https://onlinelibrary.wiley.com/doi/10.1111/j.1600-0838.2011.01299.x

OpenAI, :, Berner, C., Brockman, G., Chan, B., Cheung, V., ... Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv.org perpetual, non-exclusive license*.

Richardson, M., & Caulfield, B. (2015). Investigating traffic light violations by cyclists in dublin city centre. *Accident Analysis & Prevention*, *84*, 65–73. Retrieved 2023-07-24, from https://linkinghub.elsevier.com/retrieve/pii/S0001457515300440

Schleinitz, K., Petzoldt, T., Kröling, S., Gehlert, T., & Mach, S. (2019). (e-)cyclists running the red light – the influence of bicycle type and infrastructure characteristics on red light violations. *Accident Analysis & Prevention*, *122*, 99–107. Retrieved 2023-07-24, from https://linkinghub.elsevier.com/retrieve/pii/S0001457518307590

Tan, K. L., Poddar, S., Sharma, A., & Sarkar, S. (2019). Deep reinforcement learning for adaptive traffic signal control. *arXiv.org perpetual, non-exclusive license*.

Tilahun, N. Y., Levinson, D. M., & Krizek, K. J. (2007). Trails, lanes, or traffic: Valuing bicycle facilities with an adaptive stated preference survey. *Transportation Research Part A: Policy and Practice*, *41*(4), 287–301. Retrieved 2021-03-04, from https://linkinghub.elsevier.com/retrieve/pii/S096585640600108X

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*.

Wang, S., Xie, X., Huang, K., Zeng, J., & Cai, Z. (2019). Deep reinforcement learning-based traffic signal control using high-resolution event-based data. *Entropy*, *21*(8), 744. Retrieved 2023-02-02, from https://www.mdpi.com/1099-4300/21/8/744

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, 20–22 Jun). Dueling network architectures for deep reinforcement learning. In M. F. Balcan & K. Q. Weinberger (Eds.), (Vol. 48, pp. 1995–2003). New York, New York, USA: PMLR. Retrieved from https://proceedings.mlr.press/v48/wangf16.html

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292. Retrieved 2023-07-07, from http://link.springer.com/10.1007/BF00992698

Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). IntelliLight: A reinforcement learning approach for intelligent traffic light control. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2496–2505.

Appendix A. Variables used

Table A1. Table summarizing the variables used.

Section	Name	Value	Description
1.2	s_t	Section 3.2	State of the environment sent to the DRL agent at step t.
	a_t	Section 3.3	Action chosen by the agent at step t.
	r_t	Section 3.4	Reward quantifying the quality of a_t.
3.5.2	Memory buffer size	25000	Maximum size of DRL agent memory buffer
	Batch size	128	Size of the batches used by the DRL agent during its learning.
	Starting ϵ	1	Value of ϵ (used by the ϵ-greedy policy) at the start of training.
	Ending ϵ	0.01	Value of ϵ at the end of training.
	Pre-training acts pt	10000	Number of decisions made by the DQN agent before the first learning step.
	Final action f	1500000	Number of decisions made before the training ends.
	Discount-rate γ	0.99	Discount-rate weighting the distant future decisions made.
	Target update υ	7500	Number of decisions made between two updates of the target network.
	Learning rate	0.001	Learning rate of the Q-network.
4.2	λ_p	Section 4.2	Parameter of a Poisson process p modelling the traffic demand of a lane.
4.3	$minDur$	10	*actuated* method parameter defining the minimum number of seconds of a green phase.
	$maxDur$	40	*actuated* method parameter defining the maximum number of seconds of a green phase.
	$duration$	5	Duration of the *actuated* method's timer managing the dynamic phase change.