A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management

Andrea Vidali, Luca Crociani, Giuseppe Vizzari, Stefania Bandini
CSAI - Complex Systems & Artificial Intelligence Research Center,
University of Milano-Bicocca, Milano, Italy
name.surname@unimib.it

***Abstract*—Traffic monitoring and control, as well as traffic simulation, are still significant and open challenges despite the significant researches that have been carried out, especially on artificial intelligence approaches to tackle these problems. This paper presents a Reinforcement Learning approach to traffic lights control, coupled with a microscopic agent-based simulator (Simulation of Urban MObility - SUMO) providing a synthetic but realistic environment in which the exploration of the outcome of potential regulation actions can be carried out. The paper presents the approach, within the current research landscape, then the specific experimental setting and achieved results are described.**

***Index Terms*—reinforcement learning, traffic lights control, traffic management, agent-based simulation**

I. Introduction

Traffic monitoring and control and, in general, approaches supporting the reduction of congestion still represent hot topics for research of different disciplines, despite the substantial researches that have been devoted to these topics. The global phenomenon of urbanization (half of the world's population was living in cities at the end of 2008 and it is predicted that by 2050 about 64% of the developing world and 86% of the developed world will be urbanized[1]) is in fact constantly changing the situation and making it actually harder to manage such a concentration of population and transportation demand. Technological developments among which autonomous driving represents just the most futuristic one (at least from a popular culture perspective), represent at the same time attempts to tackle these issues and further challenges, in terms of potential developments whose introduction requires further study and analysis of the potential impact and implications.

Artificial Intelligence plays an important role within this framework; even not considering the obvious relevance to the autonomous driving initiative, we focus here on two aspects: (i) the regulation of traffic patterns, especially based on (ii) the analysis of situations by means of agent-based simulations, in which the behaviour of drivers and other relevant entities is modeled and computer within a synthetic environment. The latter, in particular, have reached a level of sufficient complexity, flexibility, and they have proven their capability to support decision makers in the exploration of alternative ways to manage traffic within urban settings. On the side of regulation of traffic patterns, the availability of these simulators, coupled with advances in machine learning, represents an opportunity for a scientific investigation of the possibility to employ these virtual environments as tools to explore the outcome of potential regulation actions within specific situations, within a Reinforcement Learning [1] framework.

This paper represents a contribution within this line of research and, in particular, we focus on a simple yet still studied situation: a single four way intersection regulated by traffic lights, that we want to manage through an autonomous agent perceiving the current traffic conditions, and exploiting the experience carried out in simulated situations, possibly representing plausibile traffic conditions. The simulations are actually also agent-based, and in particular, for this study, they have been carried out in a tool for Simulation of Urban MObility (SUMO) [2] providing a synthetic but realistic environment in which the exploration of the outcome of potential regulation actions can be carried out. An important aspect is the fact that SUMO provides an Application Programming Interface for interfacing with external programs, therefore we were able to define a *plausible* set of observable aspects of the environment, control the traffic lights according to the decisions of the learning agent, as well as also to exploit some stastics gathered by SUMO to describe the overall traffic flow and therefore to define the reward to the actions carried out by the traffic lights control agent.

The paper breaks down as follows: we first provide a compact description of the relevant portion of the state of the art in traffic lights management with RL approaches, then we introduce the experimental setting we adopted for this study. The RL approach we defined and adopted will be given in Section IV, then the achieved results will be described. Conclusions and future developments will end the paper.

II. Related Works

A. Reinforcement Learning

One of the acceptations of the goals of AI is to develop machines that resemble the *intelligent* behavior of a human being. In order to achieve this goal, an AI system should be able to interact with the environment and learn how to correctly act inside it. An established area of AI that has been proved capable of experience-driven autonomous learning is reinforcement learning [1]. Several complex tasks were successfully completed using reinforcement learning in multiple fields, such as games [3], robotics [4], and traffic signal control.

[1] https://population.un.org/wup/

In a Reinforcement Learning (RL) problem, an autonomous agent observes the environment and perceives a state s_t, which is the state of the environment at time t. Then the agent chooses an action a_t which leads to a transition of the environment to the state s_{t+1}. After the environment transition, the agent obtains a reward r_{t+1} which tells the agent how good a_t was with respect to a performance measure. The goal of the agent is to learn the policy π^* that maximizes the cumulative expected reward obtained as a result of actions taken while following π^*. The standard cycle of reinforcement learning is shown in Figure 1.

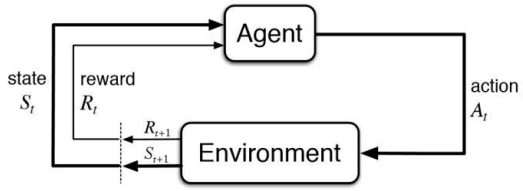

Fig. 1. The reinforcement learning cycle.

B. Learning in Traffic Signal Control

Traffic signal control is a well suited application context for RL techniques: in this framework, one or more autonomous agents have the goal of maximizing the efficiency of traffic flow that drives through one or more intersection controlled by traffic lights. The use of RL for traffic signal control is motivated by several reasons [5]: (i) if trained properly, RL agents can adapt to different situations (e.g. road accidents, bad weather conditions); (ii) RL agents can self-learn without supervision or prior knowledge of the environment; (iii) the agent only needs a simplified model of the environment (essentially related to the state representation), since the agent learns using the system performance metric (i.e. the reward).

RL techniques applied to traffic signal control address the following challenges: [5]

- **Inappropriate traffic light sequence**. Traffic lights usually choose the phases in a static, predefined policy. This method could cause the activation of an inappropriate traffic light phase in a situation that could cause an increase in travel times.
- **Inappropriate traffic light durations**. Every traffic light phase has a predefined duration which does not depend on the current traffic conditions. This behavior could cause unnecessary waitings for the green phase.

Although the above are potential advantages of the RL approach to traffic signal control, not all of them have already been achieved, and (as we will show in the remander of the paper) the present approach only represents an initial step in this overall line of work.

In order to apply a RL algorithm, it is necessary to define the state representation, the available actions and the reward functions; in the following, we will describe the most widely adopted approaches for the design of these elements within the context of Traffic Signal Control.

1) State representation: The state is the agent's perception of the environment in an arbitrary step. In literature, state space representations particularly differ in information density.

In *low information density* representations, usually the intersection's lanes are discretized in cells along the length of the lane. Lane cells are then mapped to cells of a vector, which marks 1 if a vehicle is inside the lane cell, 0 otherwise [6]. Some approaches include additional information, adopting such a vector of car presence with the addition of a vector encoding the relative velocity of vehicles [7]. The current traffic light phase could also be added as a third vector [8].

Regarding state representations with *high information density*, usually the agent receives an *image* of the current situation of the whole intersection, i.e. a snapshot of the simulator being used; multiple successive snapshots will be stacked together to give the agent a sense of the vehicle motion [9].

2) Actions representation: In the context of traffic signal control, the agent's actions are implemented with different degrees of flexibility and they are described below.

Among the category of action set with *low flexibility*, the agent can choose among a defined set of light combinations. When an action is selected, a *fixed amount of time* will lasts before the agent can select a new configuration [7]. Some works gave the agent more flexibility by defining phase duration with variable length [10]. An agent with a *higher flexibility* chooses an action at every step of the simulation from a fixed set of light combinations. However, the selected action is not activated if the minimum amount of time required to release at least a vehicle, has not passed [8], [9]. A slightly different approach would be to have a defined cycle of light combinations activated into the intersection. The agent action is represented by the choice of *when it is time to switch* to the next light combination, and the decision is made at every step [11].

3) Reward representation: The reward is used by the agent to understand the effects of the latest action taken in the latest state; it is usually defined as a function of some performance indicator of the intersection efficiently, such as vehicles' delays, queue lengths, waiting times or overall throughput.

Most of the works include the calculation of the change between cumulative vehicle delay between actions, where the vehicle delay is defined as the number of seconds the vehicles is steady [8], [9]. Similarly, the cumulative vehicle staying time can be used, which is the number of seconds the vehicle has been steady since his entrance in the environment [7]. Moreover, some works combine multiples indicators in a weighted sum [11].

C. Adopted models and learning algorithms

The most recent reinforcement learning research has proposed multiple possible solutions to address the traffic signal control problem, in which it emerges that different algorithms and neural networks structure can be used, although some common techniques are necessary but not sufficient in order to ensure a good performance.

The most widely used algorithm to address the problem is Q-learning. The optimal behavior of the agent is achieved with

the use of neural networks to approximate Q-values given a state. Often, this approach includes a Convolutional Neural Network (CNN) to compute the environment state and learn features from an image [9] or a spatial representation [8], [7].

Genders and Ravi [8] and Gao et al. [7] make use of a Convolutional Neural Network to learn features from their spatial representation of the environment. The output of this network with the current phase is passed to two fully connected layers that connect to the outputs represented by Q-values. This method showed good results in [7] work against different traffic lights policies, such as long-queue-first and fixed-times, while in [8] it is compared to a shallow neural network, in which (although it shows a good performance) an evaluation against real-world traffic lights would lead to more significant results.

Mousavi et al. [9] analyzed a double approach to address the traffic signal control problem. The first approach is value-based, while the second is policy-based. In the first approach, action values are predicted by minimizing the mean-squared error of Q-values with the stochastic gradient-descent method. In the alternative approach, the policy is learned by updating the policy parameters in such a way that the probability of taking good actions increases. A CNN is used as a function approximator to extract features from the image of the intersection, wherein the value-based approach the output is the value of actions, and in the policy-based approach it is a probability distribution over actions. Results show that both the approaches achieve good performance against a defined baseline and do not suffer from instability issues.

In [10], a deep stacked autoencoders (SAE) neural network is used to learn Q-values. This approach uses autoencoders to minimize the error between the encoder neural network Q-value prediction and the target Q-value by using a specific loss function. It is shown that achieves better performance than traditional RL methods.

III. Experimental Setting

The traffic microsimulator used for this research is Simulation of Urban MObility (SUMO) [12]. SUMO provides a software package which includes an infrastructure editor, a simulator interface and an application programming interface (API). These elements enable the user to design and implement custom configurations and functionalities of a road infrastructure and exchange data during the traffic simulation.

In this research, the chance of improvement in traffic flow that drives through an intersection controlled by traffic lights will be investigated using artificial intelligence techniques. The agent is represented by the traffic light system that interacts with the environment in order to maximize a certain measure of traffic efficiency. Given this general premise, the problem tackled in this paper is defined as follows: given the state of the intersection, what is the traffic light phase that the agent should choose, selected from a fixed set of predefined actions, in order to maximize the reward and consequently optimize the traffic efficiency of the intersection.

The typical workflow of the agent is shown in Figure 2. It should be underlined that in this application with SUMO, the passage of time is represented in simulation steps. But the agent only operates at certain steps, after the environment has evolved enough. Therefore, in this paper every step dedicated to the agent's workflow is called agentstep, while the steps dedicated to the simulation are simply called "steps". Hence, after a certain amount of simulation steps, the agent starts its sequence of operations by gathering the current state of the environment. Also, the agent calculates the reward of the previous selected action, using some measure of the current traffic situation. The sample of data containing every information about the latest simulation steps is saved to a memory and later extracted for a training session. Now the agent is ready to select a new action based on the current state of the environment, which will resume the simulation until the next agent interaction.

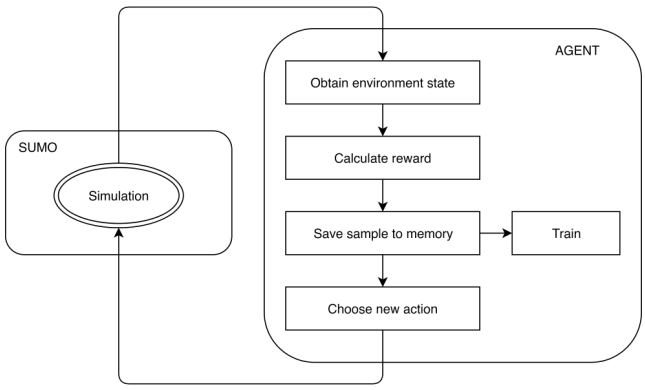

Fig. 2. The agent's workflow.

The environment where the agent acts is represented in Figure 3. It is a 4-way intersection where 4 lanes per arm approach the intersection from the compass directions, leading to 4 lanes per arm leaving the intersection. Each arm is 750 meters long. On every arm, each lane defines the possible directions that a vehicle can follow: the right-most lane enable vehicles to turn right or going straight, the two central lanes bound the driver to go straight while on the left-most lane the left turn is the only direction allowed. In the center of the intersection, a traffic light system, controlled by the agent, manages the approaching traffic. In particular, on every arm the left-most lane has a dedicated traffic light, while the other three lanes share a traffic light. Every traffic light in the environment operates according to the common european regulations, with the only exception being the absence of time between the end of a yellow phase and the start of the next green phase. In this environment pedestrians, sidewalks and pedestrian crossings are not included.

A. Training setup and traffic generation

The entire training is divided in multiple episodes .The total number of episodes is 300. By default, SUMO provides a time frequency of 1 second per step, and the period of each episode is set at 1 hour and 30 minutes, therefore the total number of steps per episode is equal to 5400. 300 episodes of 1.30 hours each are equivalent to almost 19 days of continuous traffic, and the entire training takes about 6 hours on a high-end laptop.

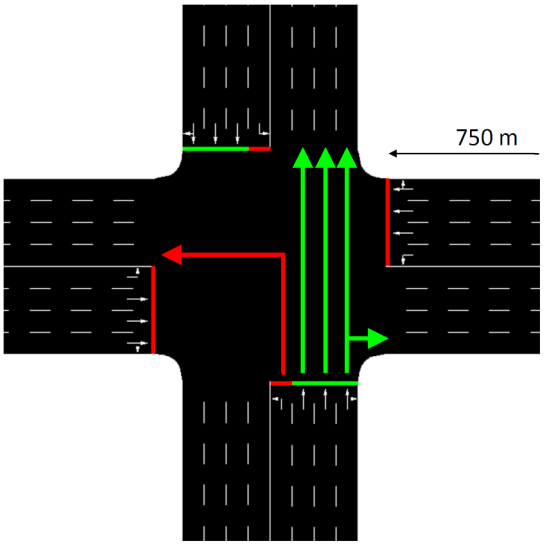

Fig. 3. The environment.

In a simulated intersection, the traffic generation is a crucial part that can have a big impact on the agents performance. In order to maintain a high degree of reality, in each episode the traffic will be generated according to a Weibull distribution with a shape equal to 2. An example is shown in Figure 4. The distribution is presented in the form of a histogram, where the steps of one simulation episode are defined on the x-axis and the number of vehicles generated in that step window is defined on the y-axis. The Weibull distribution approximates specific traffic situations, where during the early stage the number of cars is rising, representing a peak hour. Then, the number of incoming cars slowly decreases describing the gradual mitigation of traffic congestion. Also, every vehicles generated has the same physical dimensions and performance.

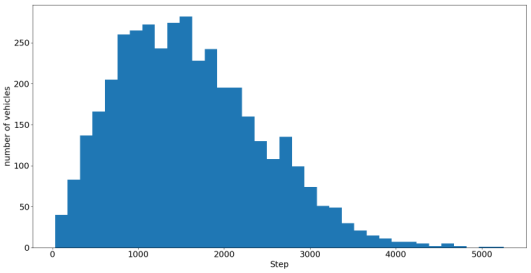

Fig. 4. Traffic generation distribution over a single episode.

The traffic distribution described provides the exact step of the episode when a vehicle will be generated. For every vehicle scheduled, its source arm and destination arm are determined using a random number generator which have a different seed in every episode, so it is not possible to have two equivalent episodes. In order to obtain a true adaptive agent, the simulation should include a significant variety of traffic flows and patterns [13]. Therefore, four different scenarios are defined and they are the following.

- High-traffic scenario. 4000 cars approach the intersection from every arm evenly distributed. Then, 75 % of generated cars will go straight and 25 % of cars will turn left or right at the intersection.
- Low-traffic scenario. 600 cars approach the intersection from every arm evenly distributed. Then, 75 % of generated cars will go straight and 25 % of cars will turn left or right at the intersection.
- NS-traffic scenario. 2000 cars approach the intersection, with 90 % of them coming from the North or South arm. Then, 75 % of generated cars will go straight and 25 % of cars will turn left or right at the intersection.
- EW-traffic scenario. 2000 cars approach the intersection, with 90 % of them coming from the East or West arm. Then, 75 % of generated cars will go straight and 25 % of cars will turn left or right at the intersection.

Each scenario corresponds to one single episode and they cycle during the training always in the same order.

IV. Description of the Reinforcement Learning Approach

In order to design a system based on the reinforcement learning framework, it is necessary to define the state representation, the action set, the reward function and the agent learning techniques involved. It should be noted that the such agent's elements in this paper are easily replaceable with a traffic monitoring system in a real world appliance, compared to others relevant studies in this topic which have higher requirements in terms of technical feasibility.

A. State representation

The state of the agent describes a representation of the situation of the environment in a given agentstep t and it is usually denoted with s_t. To allow the agent to effectively learn to optimize the traffic, the state should provide sufficient information about the distribution of cars on each road.

The objective of the chosen representation is to let the agent knows the position of vehicles inside the environment at agentstep t. For this purpose the approach proposed in this paper is inspired to the DTSE [8], with the difference that less information is encoded in this state. In particular, this state design includes only spatial information about the vehicles hosted inside the environment, and the cells used to discretize the continuous environment are not regular. The chosen design for the state representation is focused on realism: recent works on traffic signal controller proposed information-rich states, but in reality they hard to implement since the information used in that kind of representations is difficult to gather. Therefore, in this paper will be investigated the chance of obtaining good results with a simple and easy-to-apply state representation.

Technically, in each arm of the intersection incoming lanes are discretized in cells that can identify the presence or absence of a vehicle inside them. In Figure 5 is showed the state representation for the west arm of the intersection. Between the beginning of the road and the intersection's stop line, there are 20 cells. 10 of them are located along the left-only lane while the others 10 cover the others three lanes. Therefore, in

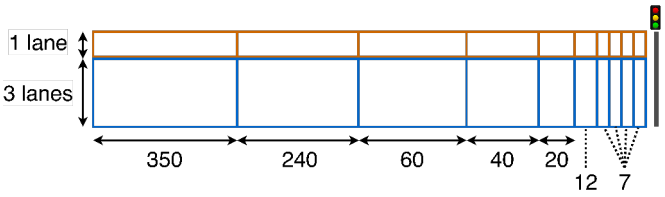

Fig. 5. Design of the state representation in the west arm of the intersection, with cells length.

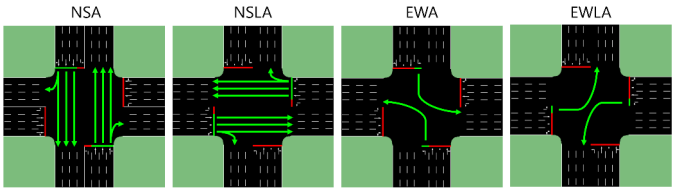

Fig. 6. Graphical representation of the four possible actions.

the whole intersection there are 80 cells. Not every cell has the same size: the further the cell is from the stop line, the longer it is, so more lane length is covered. The choice of the length of every cell is not trivial: if cells were too long, some cars approaching the crossing line may not be detected; if cells were too short, the number of states required to cover the length of the lane increases, bringing to higher computational complexity. In this paper, the length of the shortest cells, which are also the closest to the stop line, is exactly 2 meters longer than the length of a car.

In summary, whenever the agent observe the state of the environment, he will obtain the set of cells that describe the presence or absence of vehicles in every area of the incoming roads.

B. Action set

The action set identifies the possible actions that the agent can take. The agent is the traffic light system, so doing an action translates to activate a green phase for a set of lanes for a fixed amount of time, choosing from a predefined set of green phases. In this paper, the green time is set at 10 seconds and the yellow time is set at 4 seconds. Formally, the action space is defined in the set (1). The set includes every possible action that the agent can take.

$$A = \{\text{NSA}, \text{NSLA}, \text{EWA}, \text{EWLA}\} \tag{1}$$

Every action of set (1) is described below.

- North-South Advance (NSA): the green phase is active for vehicles that are in the north and south arm and wants to proceed straight or turn right.
- North-South Left Advance (NSLA): the green phase is active for vehicles that are in the north and south arm and wants to turn left.
- East-West Advance (EWA): the green phase is active for vehicles that are in the east and west arm and wants to proceed straight or turn right.
- East-West Left Advance (EWLA): the green phase is active for vehicles that are in the east and west arm and wants to turn left.

Figure 6 shows a graphical representation of the four possible actions.

If the action chosen in agentstep t is the same as the action taken in the last agentstep $t-1$ (i.e. the traffic light combination is the same), there is no yellow phase and therefore the current green phase persists. On the contrary, if the action chosen in agentstep t is not equal to the previous action, a 4 seconds yellow phase is initiated between the two actions. This means that the number of simulation steps between two same actions is 10, since 1 simulation step is equal to 1 second in SUMO. When the two consecutive actions are different, the yellow phase counts as 4 extra simulation steps and therefore the total number of simulation steps in between actions is 14. Figure 7 shows a brief scheme of this process.

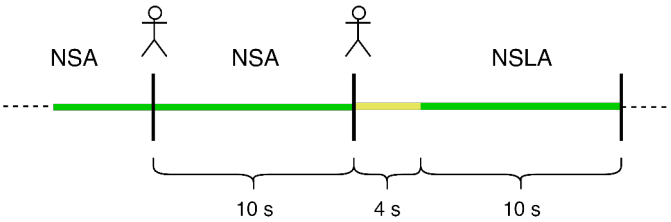

Fig. 7. Possible differences of simulation steps between actions.

C. Reward function

In reinforcement learning, the reward represents the feedback from the environment after the agent has chosen an action. The agent uses the reward to understand the result of the taken action and improve the model for future action choices. Therefore, the reward is a crucial aspect of the learning process. The reward usually has two possible values: positive or negative. A positive reward is generated as a consequence of good actions, a negative reward is generated from bad actions. In this application, the objective is to maximize the traffic flow through the intersection over time. In order to achieve this goal, the reward should be derived from some performance measure of traffic efficiency, so the agent is able to understand if the taken action reduce or increase the intersection efficiency. In traffic analysis, several measures are used [14], such as throughput, mean delay and travel time. In this paper, two reward functions are presented which use two slightly different traffic measures, and they are the following.

1) Literature reward function: The first reward function is called *literature* because it is inspired to similar studies in this topic. The literature reward function uses as a metric the *total waiting time*, defined as in equation (2).

$$twt_t = \sum_{veh=1}^{n} wt_{(veh,t)} \tag{2}$$

Where $wt_{(veh,t)}$ is the amount of time in seconds a vehicle veh has a speed of less than 0.1 m/s at agentstep t. n represents the total number of vehicles in the environment in agentstep t.

Therefore, twt_t is the total waiting time at agentstep t. From this metric, the literature reward function can be defined as a function of twt_t and is shown in (3)

$$r_t = 0.9 \cdot twt_{t-1} - twt_t \tag{3}$$

Where r_t represents the reward at agentstep t. twt_t and twt_{t-1} represent the total waiting time of all the cars in the intersection captured respectively at agentstep t and $t-1$. The parameter 0.9 helps with the stability of the training process.

In a reinforcement learning application, the reward usually can be positive or negative, and this implementation is no exception. The equation 3 is designed in such a way that when the agent chooses a bad action it returns a negative value and when it chooses a good action it returns a positive value. A bad action can be represented as an action that, in the current agentstep t, adds more vehicles in queues compared to the situation in the previous agentstep $t-1$, resulting in higher waiting times compared to the previous agentstep. This behavior increases the twt for the current agentstep t and consequently the equation 3 assumes a negative value. The more vehicles were added in queues for the agentstep t, the more negative r_t will be and therefore the worst the action will be evaluated by the agent. The same concept is applied for good actions.

The problem with this reward function lays inside the choiche of the metric, and happens when the following situation arise. During the High-traffic scenario, very long queues appears. When the agent activate the green phase for a long queue, the departure of cars creates a wave of movement that traverse the entire queue. The reward associated to this phase activation is received not only in the next agentstep, as it should, but also in very next ones. That is because the movement wave persists longer compared to the delta step between actionstep, and the wave triggers the waiting times of cars in the queue, misleading the agent about the reward received.

2) Alternative reward function: The alternative reward function uses a metric that is slightly different from the former metric, which is the *accumulated total waiting time*, defined in equation (4).

$$atwt_t = \sum_{veh=1}^{n} awt_{(veh,t)} \tag{4}$$

Where $awt_{(veh,t)}$ is the amount of time in seconds a vehicle veh has a speed of less than 0.1 m/s at agentstep t, since the spawn into the environment. n represents the total number of vehicles in the environment in agentstep t. Therefore, $atwt_t$ is the accumulated total waiting time at agentstep t. With this metric, when the vehicle departs but it does not manage to cross the intersection, the value of $atwt_t$ does not resets (unlike the value of twt_t), avoiding the misleading reward associated with the literature reward function, when a long queue build up at the intersection. Once the metric is set, the alternative reward function is defined such as in equation (5)

$$r_t = atwt_{t-1} - atwt_t \tag{5}$$

Where r_t represents the reward at agentstep t. $atwt_t$ and $atwt_{t-1}$ represent the accumulated total waiting time of all the cars in the intersection captured respectively at agentstep t and $t-1$.

D. Deep Q-Learning

The learning mechanism involved in this paper is called Deep Q-Learning, which is a combination of two aspects widely adopted in the field of reinforcement learning: deep neural networks and Q-Learning. Q-Learning [15] is a form of model-free reinforcement learning [16]. It consists of assigning a value, called the *Q-value*, to an action taken from a precise state of the environment. Formally, in literature, a Q-value is defined as in equation (6).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \cdot max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \tag{6}$$

where $Q(s_t, a_t)$ is the value of the action a_t taken from state s_t. The equation consists on updating the current Q-value with a quantity discounted by the learning rate α. Inside the parenthesis, the term r_{t+1} represents the reward associated to taking action a_t from state s_t. The subscript $t+1$ is used to emphasize the temporal relationship between taking the action a_t and receiving the consequent reward. The term $Q(s_{t+1}, a_t)$ represents the immediate future's Q-value, where s_{t+1} is next state in which the environment has evolved after taking action a_t in state s_t. The expression max_A means that, among the possible actions a_t in state s_{t+1}, the most valuable is selected. The term γ is the discount factor that assumes a value between 0 and 1, lowering the importance of future reward compared to the immediate reward.

In this paper, a slightly different version of the equation (6) is used and it is presented in equation (7). This will be called the Q-learning function from this point.

$$Q(s_t, a_t) = r_{t+1} + \gamma \cdot max_A Q'(s_{t+1}, a_{t+1}) \tag{7}$$

Where the reward r_{t+1} is the reward received after taking action a_t in state s_t. The term $Q'(s_{t+1}, a_{t+1})$ is the Q-value associated with taking action a_{t+1} in state s_{t+1}, i.e. the next state after taking action a_t in state s_t. As seen in equation (6), the discount factor γ denote a small penalization of the future reward compared to the immediate reward. Once the agent is trained, the best action a_t taken from state s_t will be the one that maximize the function $Q(s_t, a_t)$. In other words, maximizing the Q-learning function means following the best strategy that the agent have learned.

In a reinforcement learning application, often the state space is so large that is impractical to discover and save every state-action pair. Therefore, the Q-learning function is approximated using a neural network. In this paper, a fully connected deep neural network is used, which is composed of an input layer of 80 neurons, 5 hidden layers of 400 neurons each with rectified linear unit (ReLU) [17] and the output layer with 4 neurons with linear activation function, each one representing the value of an action given a state. A graphical representation of the deep neural network is showed in Figure 8

E. The training process

Experience replay [18] is a technique adopted during the training phase in order to improve the performance of the

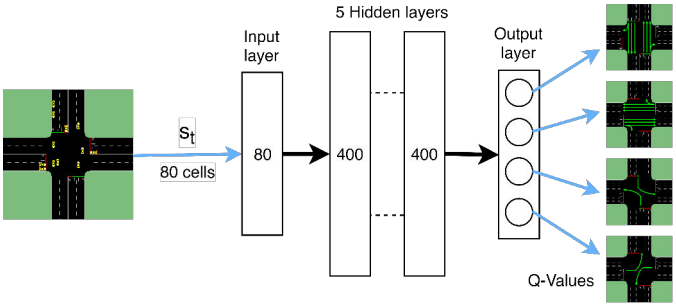

Fig. 8. Scheme of the deep neural network.

agent and the learning efficiency. It consists of submitting to the agent the information needed for learning in the form of a randomized group of samples called *batch*, instead of immediately submitting the information that the agent gather during the simulation (commonly called *Online Learning*). The batch is taken from a data structure intuitively called memory, which stores every sample collected during the training phase. A sample m is formally defined as the quadruple (8).

$$m = \{s_t, a_t, r_{t+1}, s_{t+1}\} \tag{8}$$

Where r_{t+1} is the reward received after taking the action a_t from state s_t, which evolves the environment into the next state s_{t+1}. This technique is implemented to remove correlations in the observation sequence, since the state of the environment s_{t+1} is a direct evolution of the state s_t and the correlation can decrease the training capability of the agent. In Figure 9 is shown a representation of the data collection task.

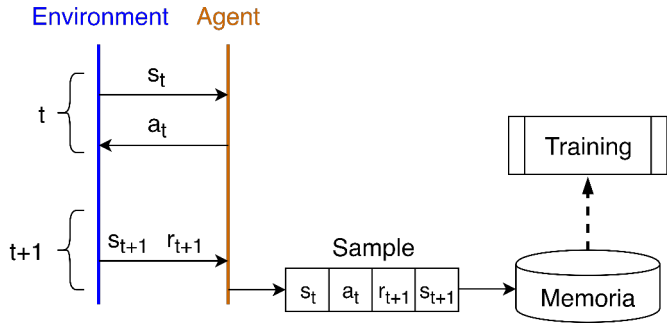

Fig. 9. Scheme of the data collection.

As stated earlier, the experience replay technique needs a memory, which is characterized by a memory size and a batch size. The memory size represents how many samples the memory can store and is set at 50000 samples. The batch size is defined as the number of samples that are retrieved from the memory in one training instance and is set at 100. If at a certain agentstep the memory is filled, the oldest sample is removed to make space for the new sample.

A training istance consists of learning the Q-value function iteratively using the information contained in the batch of samples extracted. Every sample in the batch is used for training. From the standpoint of a single sample, which contains the elements $\{s_t, a_t, r_{t+1}, s_{t+1}\}$, the following operations are executed:

1) Prediction of the Q-values $Q(s_t)$, which is the current knowledge that the agent has about the action values from s_t.
2) Prediction of the Q-values $Q'(s_{t+1})$. These represents the knowledge of the agent about the action values starting from the state s_{t+1}.
3) Update of $Q(s_t, a_t)$ which represents the value of the particular action a_t selected by the agent during the simulation. This value is overwritten using the Q-learning function described in equation (7). The element r_{t+1} is the reward associated to the action a_t, $max_A Q'(s_{t+1}, a_{t+1})$ is obtained using the prediction of $Q'(s_{t+1})$ and represents the maximum expected future reward i.e. the higher action value expected by the agent, starting from state s_{t+1}. It will be discounted by a factor γ that gives more importance to the immediate reward.
4) Training of the neural network. The input is the state s_t, while the desired output is the updated Q-values $Q(s_t, a_t)$ that now includes the maximum expected future reward thanks to the Q-value update.

Once the deep neural network has sufficiently approximated the Q-learning function, the best traffic efficiency is achieved by selecting the action with the highest value given the current state. A major problem in any reinforcement learning task is the action-selection policy while learning; whether to take exploratory action and potentially learn more, or to take exploitative action and attempt to optimize the current knowledge about the environment evolution. In this paper the *ϵ-greedy* exploration policy is chosen, and it is represented by the equation (9). It defines a probability ϵ for the current episode h to choose an explorative action, and consequently a probability $1 - \epsilon$ to choose an exploitative action.

$$\epsilon_h = 1 - \frac{h}{H} \tag{9}$$

where h is the current episode of training and E is the total number of episodes. Initially, $\epsilon = 1$, meaning that the agent exclusively explores. However, as training progresses, the agent increasingly exploits what it has learned, until it exclusively exploits.

V. Simulation Results

The performance of the agents is assessed in two parts: initially, the reward trend during the training is analyzed. Then, a comparison between the agents and a static traffic light is discussed, with respect to common traffic metrics, such as cumulative wait time and average wait time per vehicle.

One agent is trained using the literature reward function, while the other one adopts the alternative reward function. Figure 10 shows the learning improvement during the training in the Low-traffic scenario of both agents, in term of cumulative negative reward i.e the magnitude of actions' negative outcomes during each episode. As it can be seen, each agent has learned a sufficiently correct policy in the Low-traffic scenario. As the training proceeded, both agents efficiently explore the environment and learn an adequate approximation of the Q-values; then, towards the end of the training, they

try to optimize the Q-values by exploiting the knowledge learnined so far. The fact that the agent with the alternative reward function has a better reward curve overall is not a strong evidence of a better performance, since haveing two different reward functions means that different reward values are produced. The performance difference will be discussed later during the static traffic light benchmark.

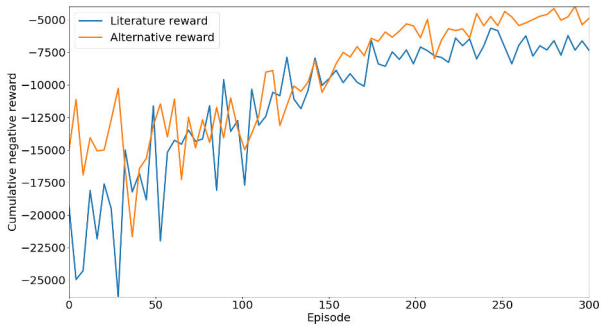

Fig. 10. Cumulative negative reward of both agents per episode during the training in the Low-traffic scenario.

Figure 11 shows the same training data as Figure 10, but referred to the High-traffic scenario. In this scenario, the agent with the literature reward shows a significantly unstable reward curve, while the other agent's trend is stable. This behavior is caused by the choiche of using the waiting time of vehicles as a metric for the reward function, which in situations with long queues causes the aquisition of misleading rewards. In fact, by using the accumulated waiting time like in the alternative reward function, vehicles does not resets their waiting times by simply advancing through the queue. As Figure 11 shows, the alternative reward function produces a more stable policy. In the NS-traffic and EW-traffic scenarios, both agents perform well since it is a simpler task to exploit.

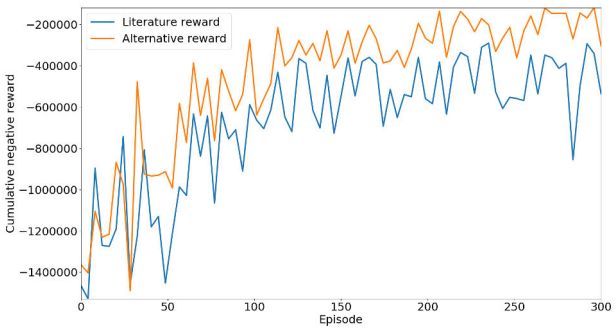

Fig. 11. Cumulative negative reward of both agents per episode during the training in the High-traffic scenario.

In order to truly analyze which agent achieve better performance, a comparison between the agents and a Static Traffic Light (STL) is presented. The STL has the same layout of the agents and it cycle through the 4 phases always in the following order: [NSA – NSLA – EWA – EWLA]. Moreover, every phase has a fixed duration and they are inspired by those on real-world static traffic lights [19]. In particular, the phases NSA and EWA lasts 30 seconds, the phases NSLA and EWLA lasts 15 seconds and the yellow phase is the same as the agent, which is 4 seconds.

In Table I are shown the performance of the two agents, compared to the STL. The metric used to measure the performance difference are the *cumulative wait time* and the *average wait time per vehicle*. The cumulative wait time is defined as the sum of all waiting times of every car during the episode, while the average waiting time per vehicle is defined as the average amount of seconds spent by a vehicle in a steady position during the episode. These measures are gathered across 5 episodes and then averaged.

	Literature reward agent	**Alternative reward agent**
	Low-traffic scenario	
cwt	-30	-47
awt/v	-29	-45
	High-traffic scenario	
cwt	+145	+26
awt/v	+136	+25
	NS-traffic scenario	
cwt	-50	-62
awt/v	-47	-56
	EW-traffic scenario	
cwt	-65	-65
awt/v	-59	-58

TABLE I
AGENTS PERFORMANCE OVERVIEW, PERCENTAGE VARIATIONS COMPARED TO STL (LOWER IS BETTER).

In general, the alternative reward agent achieves a better traffic efficiency compared to the literature agent: this is a consequence of the adoption of a reward function (accumulated waiting time) that more properly discounts waiting times *exceeding* a single traffic light cycle. Considering just the waiting time starting from the last stop of the vehicles, leads to not sufficiently emphasize the usefulness of keeping longer light cycles, introducing too many yellow lights situations and changes, that are effective in low or medium traffic situations. The fact that the agent is more effectine in low to medium traffic situations, leads to think that an easy and almost immediate opportunity would be to separately develop agents devoted to different traffic situations, having a sort of controller that monitors the traffic flow and that selects the most appropriate agent configuration. This experimentation also leads to consider that, however, additional improvements would be possible by (i) improving the learning approach to achieve a more stable and faster convergence, (ii) further improving the reward fuction to better describe the desired behaviour and to influence the average cycle lengths, that is more fruitfully short in low traffic situation and long whenever the traffic condition worsens.

VI. Conclusions and Future Developments

This work has presented a believable exploration of the plausibility of a RL approach to the problem of traffic lights adaptation and management. The work has employed a realistic and validated traffic simulator to provide an environment in which training and evaluating a RL agent. Two metrics for the reward of agent' actions have been investigated, clarifying that a proper decription of the application context is just as important as the competence in the proper application of machine learning approaches for achieving proper results.

Future works are aimed at further improving achieved results, but also, within a longer term, at investigating what would be the implications of introducing mutiple RL agents within a road network and what would be the possiblity to coordinate their efforts for achieving global improvements over local ones, and also the implications on the vehicle population, that could perceive the change in the infrastructure and adapt in turn to exploit additional opportunities and potentially negating the achieved improvements due to an additional traffic demand on the improved intersections. It is important to perform analyses along this line of work to understand the plausibility, potential advantages or even unintended negative implications of the introduction in the real world of this form ofself-adaptive system.

References

[1] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[2] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo – simulation of urban mobility: An overview," in *SIMUL 2011*, S. . U. of Oslo Aida Omerovic, R. I. R. T. P. D. A. Simoni, and R. I. R. T. P. G. Bobashev, Eds. ThinkMind, October 2011. [Online]. Available: https://elib.dlr.de/71460/

[3] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II," https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

[4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv: 1806.10293*, 2018.

[5] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 34, 2017.

[6] W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," *Procedia computer science*, vol. 130, pp. 26–33, 2018.

[7] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv:1705.02755*, 2017.

[8] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[9] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, 2017.

[10] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[11] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2496–2505.

[12] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[13] L. A. Rodegerdts, B. Nevers, B. Robinson, J. Ringert, P. Koonce, J. Bansen, T. Nguyen, J. McGill, D. Stewart, J. Suggett *et al.*, "Signalized intersections: informational guide," Tech. Rep., 2004.

[14] R. Dowling, "Traffic analysis toolbox volume vi: Definition, interpretation, and calculation of traffic analysis tools measures of effectiveness," Tech. Rep., 2007.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[17] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal- diffference learning with function approximation," in *Advances in neural information processing systems*, 1997, pp. 1075–1081.

[18] L.-J. LIN, "Reinforcement learning for robots using neural networks," *Ph.D. thesis, Carnegie Mellon University*, 1993.

[19] P. Koonce and L. Rodegerdts, "Traffic signal timing manual." United States. Federal Highway Administration, Tech. Rep., 2008.