Received June 15, 2021, accepted July 5, 2021, date of publication July 12, 2021, date of current version July 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096666

Traffic Light Control Using Hierarchical Reinforcement Learning and Options Framework

DIMITRIUS F. BORGES, JOÃO PAULO R. R. LEITE, EDMILSON M. MOREIRA, AND OTÁVIO A. S. CARPINTEIRO
Institute of Systems Engineering and Information Technology, Federal University of Itajubá, Itajubá, Minas Gerais 1303, Brazil
Corresponding author: João Paulo R. R. Leite (joaopaulo@unifei.edu.br)

ABSTRACT The number of vehicles worldwide has grown rapidly over the past decade, impacting how urban traffic is managed. Traffic light control is a well-known problem and, although an increasing number of technologies are used to solve it, it still poses challenges and opportunities, especially when considering the inefficiency of the popular fixed-time traffic controllers. This study aims to apply Hierarchical Reinforcement Learning (HRL) and Options Framework to control a signalized vehicular intersection and compare its performance with that of a fixed-time traffic controller, configured using the Webster Method. HRL combines the ability to learn and make decisions while taking observations from the environment in real-time. These capabilities bring a significant adaptive power to a highly dynamic problem. The test scenarios were built using the SUMO simulation tool. According to our results, HRL presents better performance than those of its own isolated sub-policies and the fixed-time model, indicating a simple and efficient alternative.

INDEX TERMS Intelligent systems, machine learning, reinforcement learning, simulation, traffic control.

I. INTRODUCTION

Brazilian vehicle fleet grew by 21.5% between 2012 and 2017 [1]. This fact leads to a constriction of traffic lights, especially in large cities. Similar situations occur in other parts of the globe [2]. The excess of vehicles turns intersections into bottlenecks, reducing traffic flow, increasing the average waiting time in line and, thus, the dissatisfaction of drivers. This fact is enhanced by a characteristic of most traffic lights: fixed operating times, which makes them unable to consider the traffic variation throughout the day and handle unexpected traffic bursts. Brazilian traffic guidelines [3] establishes the Webster's Formula [4] to calculate green times for intersections, using static information such as peak vehicle flow, maximum supported flow and lane width. However, it does not consider an intrinsic characteristic of the traffic: variability. Therefore, its performance decreases considerably when dealing with randomness or events unknown at the time of their configuration.

Nonetheless, the vehicular traffic ecosystem can be described through data collected, mapped, and transposed to mathematical models. For example, the dynamics of vehicle flow on the streets can be modeled using Queuing Theory [5], [6], Petri Diagrams [7] or Graph Theory [8], and the full range of tools associated with such theories can be explored to solve problems in urban traffic, especially those related to signaled intersections. Also, the application of Artificial Intelligence (AI) models, able to recognize patterns and generalize from the observation of historical data [9] and environment data, presents itself as a viable path. Artificial Intelligence and Machine Learning (ML) are areas of study already known to solve traffic control problems [10], with proposals dating from the 1980s [11].

In the AI/ML scenario, Reinforcement Learning (RL) [12] presents itself as a promising alternative, for two reasons: 1) it does not have, in its directives, the need to know in advance the complete model of the problem to which it is intended to solve and; 2) it is able to implement an *online* approach [12], that is, to gather knowledge while the agent is in operation, enabling it to learn even when faced with situations that differ greatly from those observed at the initial training phase. In addition, although RL has already been used to control traffic lights [13]–[15], there is no knowledge of the use of Hierarchical Reinforcement Learning (HRL) [16] and Options Framework [17] for modeling and decision making in the same scenario [18]. This variation of RL adds to the basic model the possibility to shift the agent's behavior according to what is observed in the environment.

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Alatas.

Our objective is to propose an AI-based model for traffic light control, and compare its performance to that of the fixed-time model. The proposed model uses intelligent agents that acquire knowledge from historical data [19] and the environment itself. It is able to recognize patterns at a traffic intersection, adapt itself to the scenario, and make decisions that favor the reduction of waiting time in line and the increase in vehicle flow. Based on previous studies [10], [20], we have chosen a model based on HRL and Options Framework, which have never been applied to this context.

The rest of this paper is divided into four sections: Section II presents the theoretical basis of the work. In Sections III and IV, the simulation model and the observed results are presented. Finally, Section V presents our conclusions.

II. THEORETICAL BACKGROUND

A. WEBSTER METHOD

Following the Brazilian Traffic Light Manual recommendations [3], green times for each phase of a traffic light are calculated using the Webster Method [4]. This method seeks to calculate the total effective green time (G_n) and the optimal cycle time (C_O) which keeps the flow at the intersection in order to minimize waiting time for the vehicles. The optimal cycle time is defined by (1), where L is the total lost time per cycle (sec) and Y is the sum of saturation flow ratios of individual lanes.

$$C_O = \frac{1.5 \cdot L + 5}{1 - Y} \tag{1}$$

L is given by the sum of lost times at each lane, which is the time during which an intersection is not used effectively by any movement. Also, as shown by (2), the saturation ratio of each lane (y_n) is calculated by the relationship between peak vehicle flow (P_n) and maximum flow supported by that lane (F_n), directly related to its width [3].

$$y_n = \frac{P_n}{F_n} \tag{2}$$

Finally, the effective green time (G_n) for each phase is given by (3).

$$G_n = \frac{y_n}{Y} \cdot (C_O - L) \tag{3}$$

B. REINFORCEMENT LEARNING AND Q-LEARNING

Reinforcement Learning enables an intelligent agent to learn on-the-fly using temporal learning in a trial-and-error manner as time goes by (t $= 1, 2, \ldots$) without the need for external supervision [17], following a Markov Decision Process (MDP) [21]. Q-learning (QL) is a RL approach in which the agent "learns" an optimal policy by experiencing the consequences of its actions, without the need to know the dynamics of the problem in advance [22]. This fact makes QL a *model-free* approach, opposite to *model-based*, which seeks to know the complete dynamics, that is, the transition probabilities and destinations for each state and action of the model [12].

This is how RL works: at time t, an agent i observes its Markovian (or memory-less) decision-making factors, or states $s_t^i \in S^i$, in the dynamic and stochastic operating environment, takes an action $a_t^i \in A^i$, and receives positive or negative consequences at the next time instant, $t + 1$. As time goes on, the agent explores all state-action pairs (s_t^i, a_t^i) and ranks them according to a Q-value. The Q-value of a state-action pair represents the effectiveness of taking action a_t^i in state s_t^i in the long term. Q-values are maintained and updated in a Q-table with $|S^i| \times |A^i|$ through (4).

$$\begin{aligned} Q_{t+1}^i(s_t^i, a_t^i) \leftarrow & (1-\alpha)Q_t^i(s_t^i, a_t^i) \\ & + \alpha[r_{t+1}^i(S_{t+1}^i) + \gamma \max_{a \in A} Q_t^i(s_{t+1}^i, a)] \end{aligned} \tag{4}$$

Equation (4) shows that the new Q-value is part of its current value and part of the new value obtained in the iteration, represented by the reward returned by the action taken (r_{t+1}^i) plus the best possible expected future, selected in a greedy way ($\max Q_t^i$). Therefore, the algorithm always takes steps, limited by $0 < \alpha \leq 1$, towards an optimal value, while watching the possible future states and rewards, limiting how further it looks by $0 < \gamma < 1$.

Each agent i goes through the following steps: 1) observes the current state; 2) selects an action greedily, by looking at the Q-table; 3) gets the reward; and 4) updates Q-value using (4). As an example [20], a traffic signal controller at an intersection (agent) observes at a time instant t the queue size of its lanes (the state) and decides to split the current traffic phase (an action). At the next time instant ($t + 1$), it realizes the consequence of its action, observing the waiting time of vehicles (the reward). The reward is then used to calculate the new expected value for this pair of state and action, while watching which state the environment is now. In short, the agent takes actions seeking to maximize the rewards and, by maximizing them, ensures that it will achieve its objective. Owing to the dynamic properties of the state, learning is crucial, once the action that maximizes the rewards for a particular state varies with time.

C. HIERARCHICAL REINFORCEMENT LEARNING

An Hierarchical Reinforcement Learning (HRL) model decomposes a RL problem into sub-problems, or sub-objectives, where the agent, which has a macro task, assumes sub-objectives at key moments to facilitate and expedite its fulfillment [16], [23]. In other terms, the agent has a policy, which seeks to achieve the main objective and selects partial policies (or sub-policies), which seek to achieve local objectives that are part of the whole, in a divide-and-conquer strategy. There are several HRL approaches in the literature, such as *Feudal Learning* [24], *MAXQ* [25] and *Options* [26], the latter being the main tool used in this study.

The Options framework extends RL by introducing temporally extended actions called *skills* or *options*. In other words, *options* are higher-level actions that extend over several time steps, generalizing MDPs to semi-Markov decision processes (SMDPs). Traditionally, *options* which are able to take moving agents to bottleneck states are sought after [27]. Bottleneck states are those which connect different densely connected regions of the state space (e.g., doorways). They have been shown to be very efficient for planning, because these states are the most frequently visited when considering the shortest path between any two states in an MDP [28].

III. SIMULATION

The open-source simulation tool SUMO [29], [30] was used. It is capable of simulating large-scale continuous urban traffic in detail, allowing the inclusion of different types of vehicles, pedestrians, traffic lights, vehicle detectors, etc. SUMO is extensively used in works which focus on vehicular traffic management [31]–[33]. The graphical interface used was TraCI [34], in its Python version.

As our purpose of is to compare the performance of different models in relation to the total flow of vehicles and their average waiting times in line, we chose to simulate a scenario that, although simple, is representative of frequent real cases: two streets that cross each other perpendicularly, both two-way, which totals four lanes. The traffic light has two phases (vertical/horizontal), as shown in Figure 1. Such composition encompasses the most common case of intersections with traffic lights, which occurs in both small and large cities.

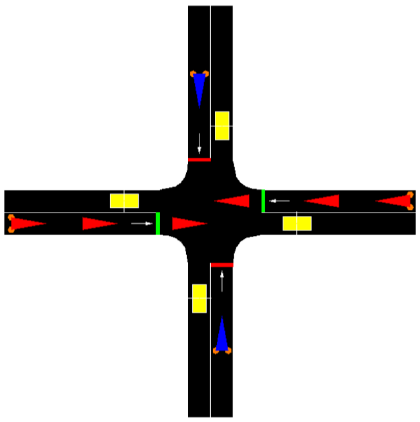


FIGURE 1. Simulated intersection in the SUMO tool. Red and blue triangles represent vehicles, yellow rectangles are vehicle detectors, green and red dashes represent traffic lights.

All parameter values are tool defaults: lanes are 3.2 meters wide, 250 meters long and have a maximum speed of 50 km/h. Vehicles are all of "passenger" type, which represent passenger cars 5 meters long, with 2.5 meters distancing between them and maximum speed of 11.11 m/s (40 km/h). The duration of the simulation is one hour, enough to observe the impacts of demand surges and traffic light management on the simulated environment. The flow of vehicles generated in the simulator is controlled by a vehicle/minute factor (v/m), by lane, and can be both deterministic, which guarantees equality between N executions, and probabilistic, obeying a Poisson distribution [35]. All scenarios used in this work made use of the deterministic generator, as this ensures total equality in the numbers and times of vehicles generated between N executions.

Two stress scenarios were elaborated: Peak-2700 and Peak-3600. They have short-term demand surges, seeking to simulate sudden changes that would not be considered when calculating times for fixed-time traffic lights (FT). In addition to them, two others scenarios were also elaborated: Fixed-1800 and Fixed-3600. Fixed-1800 represents the base case of expected flow, with no saturation or sudden change in demand. It is used to perform the time calculations for FT. Fixed-3600 is used both to saturate the intersection to the point of constriction, in alternating lanes, and in the training of the HRL agent. The four scenarios are detailed below:

- *Fixed-1800:* 1,800 vehicles generated in one hour, divided equally among all lanes.
  1) 15 v/m in each lane, for 60 minutes;
- *Fixed-3600:* 3,600 vehicles generated in one hour, divided into 3 stages:
  1) 30 v/m in each lane, for 20 minutes;
  2) Horizontal lanes with 6 v/m each, vertical lanes with 24 v/m each, for 20 minutes;
  3) Horizontal lanes with 24 v/m each, vertical lanes with 6 v/m each, for 20 minutes;
- *Peak-2700:* 2,700 vehicles generated in one hour, divided into 6 stages:
  1) 15 v/m in each lane, for 20 minutes;
  2) Horizontal lanes with 7.5 v/m each, vertical lanes with 15 v/m each, for 5 minutes;
  3) Horizontal lanes with 7.5 v/m each, vertical lanes with 45 v/m each, for 5 minutes;
  4) 15 v/m in each lane, for 20 minutes;
  5) Horizontal lanes with 15 v/m each, vertical lanes with 7.5 v/m each, for 5 minutes;
  6) Horizontal lanes with 45 v/m each, vertical lanes with 7.5 v/m each, for 5 minutes;
- *Peak-3600:* 3,600 vehicles generated in one hour, divided into 6 stages:
  1) 15 v/m in each lane, for 20 minutes;
  2) Horizontal lanes with 7.5 v/m each, vertical lanes with 45 v/m each, for 5 minutes;
  3) Horizontal lanes with 7.5 v/m each, vertical lanes with 60 v/m each, for 5 minutes;
  4) 15 v/m in each lane, for 20 minutes;
  5) Horizontal lanes with 45 v/m each, vertical lanes with 7.5 v/m each, for 5 minutes;
  6) Horizontal lanes with 60 v/m each, vertical lanes with 7.5 v/m each, for 5 minutes;

IV. EXPERIMENTS AND RESULTS

A. FIXED-TIME MODEL

Fixed-time models can assume different configurations, but remain restricted to predictable scenarios. They can, for instance, behave differently during 5-7pm, to accommodate the expected increased traffic for this period. However, they cannot automatically adapt themselves during the day if a random surge occurs, since they do not have autonomy or any input from the environment.

Flow surge predictions are not included in the FT setup. In all scenarios, including Peak-2700 and Peak-3600, vehicle flow is always set to 1800 vehicles/hour. FT has, then, two phases – vertical and horizontal –, each one with 15s of green, 3s of amber and 2s of red time. Of course, throughout the whole cycle of one phase, the other remains red. The effective cycle time is, then, 40 seconds ($(15+3+2)*2$). These values are obtained by the application of Webster Method.

As the simulator guarantees equality between executions (deterministic) and FT has a strictly fixed behavior, all executions of the scenario generate the same results. For this reason, only one round was enough to obtain the results shown in Table 1. In the table, V_s is the total number of vehicles served by the traffic light at the intersection, V_f, the average vehicle flow per minute, and W, the average waiting time for vehicles passing through the intersection.

TABLE 1. Fixed-time test scenario: results.

Scenario	V_s	V_f	W
Fixed-1800	1784	29.73	16.63
Peak-2700	2349	39.15	29.34
Peak-3600	2420	40.33	31.53

In the base case (Fixed-1800), FT is able to deliver an average waiting time of 16.63 seconds, close to the green time (15 secs), which implies that the configured time, using the Webster Method, is sufficient. It is also observed that the flow rate (29.73 v/m) is very close to the absolute rate (30 v/m, or 15 v/m for each lane).

However, in scenarios with traffic bursts, while the total number of vehicles entering the intersection increases 33% from Peak-2700 to Peak-3600, the total number of vehicles served (V_s) and the flow rate (V_f) both increase by only $\sim$3%. From these results, it is possible to conclude that the Webster Method reaches its performance limit in Peak-2700. Regardless of the volume of vehicles trying to access the intersection, the flow rate changes very little, while the waiting time continues to deteriorate. Figure 2 shows the waiting time, minute by minute, for Fixed-1800 and Peak-2700 scenarios. At Peak-2700, it is possible to verify that the waiting time begins to rapidly deteriorate at 20 minutes, right after the first flow burst.

B. HIERARCHICAL REINFORCEMENT LEARNING

1) POLICIES AND SUB-POLICIES

Two objectives were established for the HRL model, each forming a sub-policy: 1) maximizing vehicle flow (Q-Size); and 2) minimizing waiting times (W-Size). The determination of which sub-objective to pursue is made by the main policy (H-Agent). Each sub-policy is a complete *Q-learning* process that could be used individually. The H-Agent could alternately take any of them as the main policy and interact with the environment accordingly. Therefore, it is possible not only to compare the results obtained by HRL and FT models, but also with agents that use Q-Size or W-Size sub-policies exclusively.

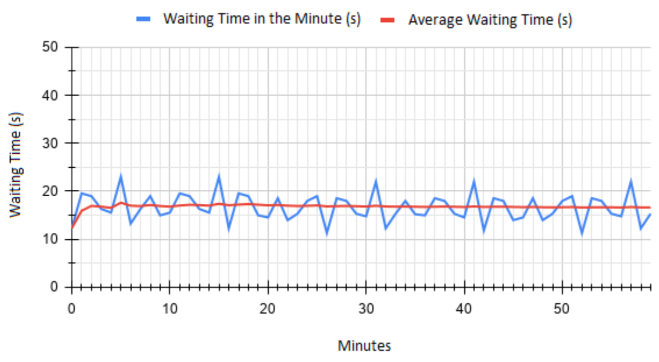
It is necessary to establish three sets of data: S, of *characteristics*, which define the states; A, of *primitive actions*, which interact for $t = 1$ instant of time with the environment; and O, of *options*, which are extended actions that interact with the environment for $t = n$ periods of time. The states are defined from an observation of the environment: 1) the number of vehicles on each lane (maximum of 25); 2) the moment each vehicle entered the queue; and 3) the phase (green or red) of each traffic light (red-amber times are ignored). The green time was set in the minimum allowed (10 seconds) [3], and the red-amber time, in 5 seconds (amber + all-red time), totalling a phase time of 15 seconds. Therefore, three types of classification were defined:

- E_q: by the number of vehicles in the queues (lanes NS/SN and WE/EW as single queues):
  - LOW: number of vehicles in the queue is less than or equal to the max flow ($\leq$20);
  - MID: number of vehicles in the queue is less than or equal to twice the max flow ($\leq$40);
  - HIGH: number of vehicles in the queue is greater than twice the max flow ($>$40).
- E_w: by the average waiting time in the queues:
  - LOW: average waiting time is less than or equal to the minimum green time ($\leq$10);
  - MID: average waiting time is less than or equal to the minimum green time + one phase time ($\leq$25);
  - HIGH: average waiting time is greater than the minimum green time + phase time ($>$25).
- E_{tf}: by the traffic light phase status:
  - g: phase is in the green state;
  - r: phase is in the red state.

The E_q flow values relate to the maximum flow in a minimum green time. To obtain them, FT was applied, with green time of 10 seconds and 5 seconds of red-amber time, to the Peak-3600 scenario.[1] The value of maximum flow in one phase obtained during this simulation was 20 vehicles. The MID and HIGH classes were determined according to the cycle definition. As a cycle is the sequence of activation of all phases and there are two, a lane with 40 vehicles consumes up to one cycle to be emptied, and is classified as MID. With more than 40 vehicles, more than one cycle is required, and the classification is HIGH.

A similar process was carried out to calculate E_w. The lane receives MID classification if the waiting time of its vehicles is, at most, 25 seconds, which indicates that the opposite lane was selected twice in a row to receive the green color (10 seconds of green +10 seconds of green +5 seconds of red-amber). This scenario is seen in Figure 3, between S_0 and S_2, where the vertical leg (V) is selected twice in a row ($a_0 = v$ and $a_1 = v$). Any value above this indicates that the

[1]Peak-3600 scenario generates flow above the capacity of the roads at moments of traffic burst.



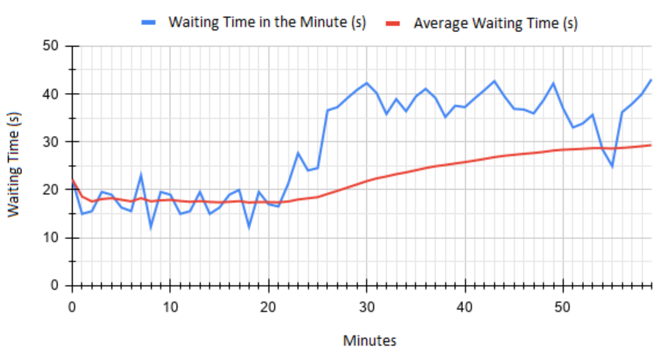


FIGURE 2. Waiting time for FT at fixed-1800 (left) and at Peak-2700 (right).

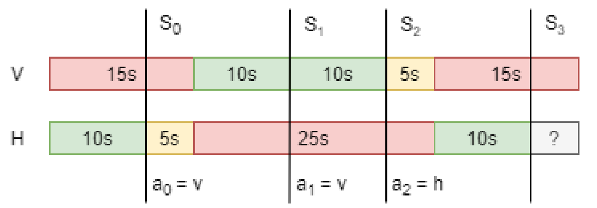


FIGURE 3. State diagram and actions over time.

lane is waiting for more than two phases (>1 cycle), leading it to HIGH classification.

Policies use combinations of the classification values defined above to describe their status. These policies will be detailed in their respective sub-topics. The primitive actions were limited to two:

- *h:* move (or keep) horizontal lanes to green state for 10 seconds;
- *v:* move (or keep) vertical lanes to green state for 10 seconds.

An action takes a minimum of 10 seconds, if it extends an already active green, or 15 seconds (5 seconds of red-amber +10 seconds of green), if it selects a phase that is currently red. The actions and, consequently, the state definitions, always occur at the end of the minimum green time of the phase that was selected in the previous action. Despite the time variation between actions, the agent always sees them as a single step, as a single interaction with the environment was made.

Figure 3 illustrates how three interactions with the environment would be. At the beginning, the agent waits until the end of the green time that is active to define the first state, S_0, and take the first action, a_0. As the first action called for an inversion of the active phase, the traffic light first activates the red-amber time and only then sets the vertical phase to green, causing a_0 to last 15 seconds. At the end of the green time, the second state definition (S_1) and action (a_1) take place, which keeps the active phase green for another 10 seconds. This action (a_1) lasts only 10s, since it is not necessary to activate the red-amber time (5s).

The *options* initiate one of the two sub-objectives to be pursued:

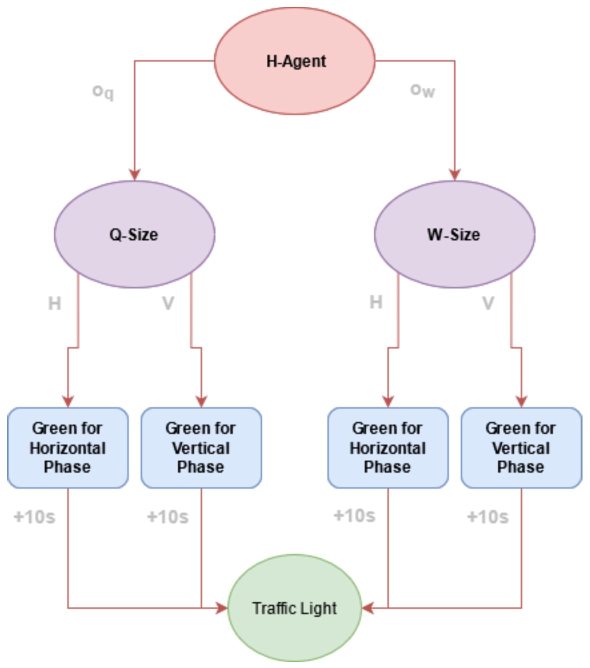


FIGURE 4. Relationship between policies and sub-policies from the HRL agent.

- σ_q: start the execution of primitive actions, following Q-Size sub-policy;
- σ_w: start the execution of primitive actions, following W-Size sub-policy.

Figure 4 summarizes the relationship between policies, options and actions. The H-Agent macro policy selects which sub-objective to pursue (maximize flow or minimize waiting time) by selecting one of the two available sub-policies (Q-Size or W-Size). The selected sub-policy performs primitive actions that directly influence the traffic light at the intersection, aiming at its own sub-objective.

2) TRAIN, TEST AND HYPERPARAMETERS

The training process was divided into three – of the main policy and of its two sub-policies. It took place through 24 runs, which simulate the duration of a full day operation of the Fixed-3600 scenario. This scenario was selected because it offers several levels of saturation on all lanes, allowing the intelligent agent to face as many traffic variations as possible. The *action*/*option* selection policy used was ϵ-greedy [36], which alternates between exploration and exploitation moments according to the ϵ parameter. The parameter represents the probability of the agent to explore its alternatives instead of selecting the best known *option*/action, which gives randomness to its behavior. Moreover, a decay logic was used, so that, at the beginning of the training, the exploration rate is higher, set at 45% ($\epsilon = 0.45$). Thereafter, for every three executions, ϵ decays by 5%, ending the training phase with an exploration rate of 10%. The test scenarios used were the same as those applied to the Fixed-Time model: Fixed-1800, Peak-2700 and Peak-3600. Tests were performed in a single execution of the scenario, with a fixed exploration rate set at 2% ($\epsilon = 0.02$). The learning rate (α) and the discount factor (γ) were individually configured for each of the three policies of the agent, and the values that showed the best performance for each of them were obtained through an exhaustive search between all possible combinations of two sets, one for α, from 0.1 to 1, with an interval of 0.1, and another for γ, from 0.1 to 0.9, also with 0.1 range.

3) Q-SIZE AND W-SIZE

In a first step, we investigate the system behavior if it had a single objective – maximizing flow (Q-Size) or minimizing waiting time (W-Size). Results will be later compared to those of the hybrid approach (H-Agent), which switches between sub-objectives according to the environment input.

E_q classification, which categorizes the vehicle queues according to their size, was used as the Q-Size sub-policy. In addition, the agent needs to know the current state of the traffic light, determined by E_{tf}. Thus, the 18 states of the Q-Size sub-policy are described by the combination of 9 queue states – LOW, MID and HIGH for each lane – and 2 traffic light states – green or red, mutually exclusive. The only set of actions available during the sub-policy is that of primordial actions A. Thus, the table $Q(s, a)$ has 18 rows and two columns, totalling a universe of 36 (s, a) pairs. Also, a reward function r, which generates feedback and indicates whether the actions are adjusted to its objective, needs to be defined. For the Q-Size policy, function r is set to 1 if the queue size decreases from moment t to $t + 1$, and set to 0, otherwise. The learning rate (α) and the discount factor (γ) were set to 0.1 and 0.4, respectively.

W-Size sub-policy, on the other hand, uses E_w classification. The agent selects that lane in which vehicles are waiting the longest to be released. Again, there are 18 states for the W-Size sub-policy, described by the combination of 9 queue states – LOW, MID and HIGH – and 2 traffic light states. As W-size is also restricted to primitive actions, $Q(s, a)$ totals, once again, 36 (s, a) pairs, with 18 states and 2 actions. The reward function r is set to 1 if the waiting time decreases from t to $t + 1$ and to 0, otherwise. The learning rate (α) and the discount factor (γ) were both set to 0.1.

Q-Size and W-Size were subjected to the same scenarios as FT: Fixed-1800, Peak-2700 and Peak-3600. However, the results are the average of 10 test/training cycles. This is due to the nature of the agent, i.e., according to its parameter ϵ, the results may differ between executions. The observed results are shown in Table 2, with the best ones highlighted in bold.

TABLE 2. W-Size, Q-Size and fixed-time summarized results.

Scenario	V_s	V_f	W
W-Size			
Fixed-1800	1782	29.23	17.14
Peak-2700	2396	39.93	**19.05**
Peak-3600	2826	47.03	**23.21**
Q-Size			
Fixed-1800	1779	29.66	25.09
Peak-2700	**2433**	**40.56**	25.32
Peak-3600	**2929**	**48.83**	27.79
Fixed-Time			
Fixed-1800	**1784**	**29.73**	**16.63**
Peak-2700	2349	39.15	29.34
Peak-3600	2420	40.33	31.53

In Fixed-1800, all approaches performed similarly, with a slight advantage for the FT agent. In this case, as the intersection flow is below its saturation point and equally distributed between the lanes, the process is trivial. Regarding the average waiting time, the Q-Size sub-policy has about 50% (~8 seconds) less performance when compared to those of the others. The lower performance is justified by the fact that the Q-Size agent only seeks to increase the flow, prioritizing the queue with more vehicles, which leads to an increasing waiting time on the opposite lane. Results show that the W-Size agent is capable to better balance its choices, obtaining results very similar to those of FT.

In the overload scenarios, the intelligent agents demonstrate their superiority. The Q-Size agent obtains the best flow values. It is 3.57% and 21% more efficient in relation to FT in Peak-2700 and Peak-3600, respectively. In short, Q-Size was able to serve more vehicles during the simulation than the other two approaches. On the other hand, the W-Size sub-policy, which focus on optimizing the waiting time, shows the expected behavior, producing waiting times 35% lower than those of FT already at Peak-2700 (24.76% lower than Q-Size as well), and 26.38% lower than those of FT at Peak-3600. Such results are in line with the proposal of each sub-policy, and indicate that a hybrid approach, capable of switching between sub-policies, is promising.

Figures 5 and 6 show, minute by minute, FT, Q-Size and W-Size performances in Peak-2700. In Figure 5, it is possible to notice that the two intelligent models are capable of meeting all the demand of the first traffic burst (at 20min) before the 40th minute of the simulation, returning to the previous flow level. The same does not occur during the

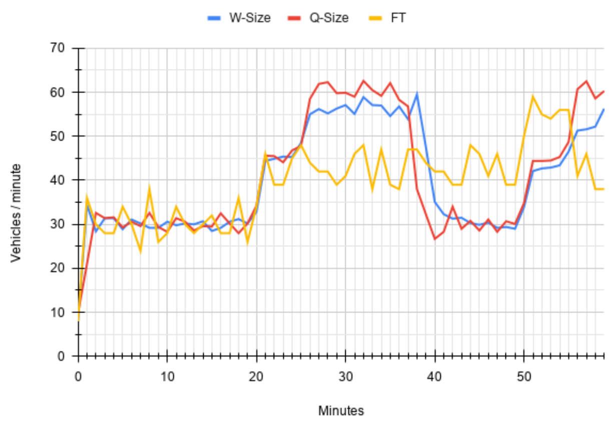


FIGURE 5. **Flow results of W-Size and Q-Size in Peak-2700.**

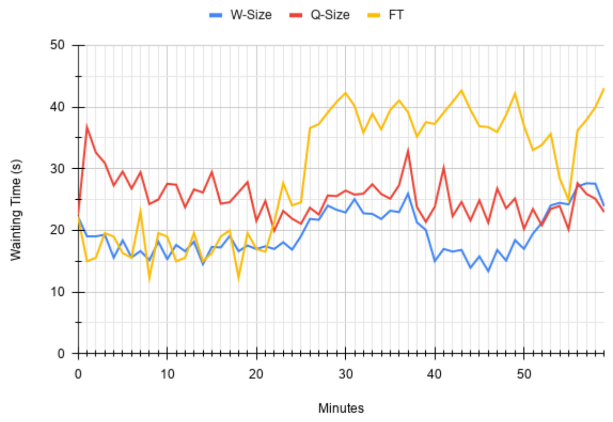


FIGURE 6. **Waiting time results of W-Size and Q-Size in Peak-2700.**

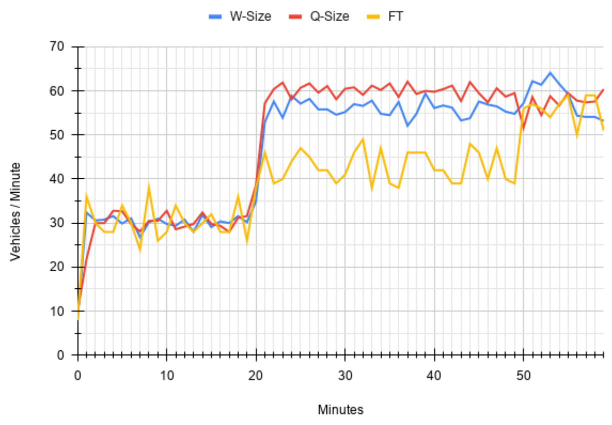


FIGURE 7. **Flow results of W-Size and Q-Size in Peak-3600.**

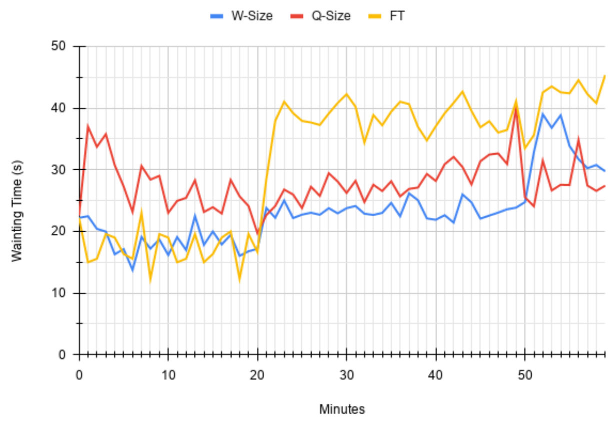


FIGURE 8. **Waiting time results of W-Size and Q-Size in Peak-3600.**

simulation with the FT approach. Also, Q-Size is capable of maintaining a higher flow at peak times (minutes 20-32 and 50-60). In Figure 6, regarding the W-Size agent, peaks of waiting time only occur when there is a burst in demand. Q-Size agent keeps a high value during the whole simulation, oblivious to the demand variations. The FT approach peaks after 20 minutes and is not able to recover its previous level until the end of the simulation.

The Peak-3600 scenario, illustrated in Figures 7 and 8, brings more accentuated differences between the agents. In Figure 7, after the first traffic burst, W-Size has its flow around 55 v/m, as can be seen from minute 21 to 50, while Q-Size manages to keep it close to 60 v/m in the same period. In the 23rd minute, Q-Size reaches its highest flow rate per minute (62 v/m). In its turn, W-Size, although reaching a higher value (64 v/m), only reaches it at 53 minutes of simulation. Although both policies reach similar average values at the end of the simulation (47.03 against 48.83), the peaks and consistency at key moments guarantee the difference in final performance. It is also worth remembering that in this scenario, there are times when the input flow is above the capacity of the road, such as, for example, in the second half of the first surge, with 60 v/m in each lane. However, Q-Size still can achieve values greater than W-Size.

In Figure 8, W-Size exhibits a behavior quite different from that that was observed previously in relation to waiting times. It is possible to see how the waiting times start low, then rise and remain so after the first traffic burst, unlike those that were seen in Figure 6, in which the variations followed the beginning and end of the burst. This is due to the fact that the number of vehicles on the lane exceeds the minimum green flow capacity, which leads to an increase in the waiting time and, as the flow is greater, there is no moment of relief. However, as the agent is guided to decrease such values, it is able to ensure that the value does not rise sharply, as it occurs with FT, preventing frequent losses of performance.

It is possible to notice that each individual policy presents its pros and cons in relation to the other. The W-size agent obtains lower waiting times at the cost of lower vehicle flow (Figures 5 and 7). More clearly, the Q-size agent achieves higher values for vehicle flow, but it has a lower performance regarding the waiting times, mainly before the first traffic burst (Figures 6 and 8). If any lane has a bigger waiting time but contains fewer vehicles, it will be ignored by the Q-agent, possibly increasing the waiting time. In short, both agents observe only the variable that is important for their objectives, neglecting the other.

4) H-AGENT POLICY

The H-Agent, governed by the main policy, seeks to optimize both vehicle flow and waiting times and, for this, makes use of the two sub-policies to achieve its goal. The main policy selects one of the sub-policies at the time in which this sub-policy will have an optimal return on its objective. The H-Agent uses, as state, the combination of useful states for each sub-policy – E_q, from Q-Size, and E_w, from W-Size. The main policy must observe the environment in the same way as the sub-policies to be able to identify which one will perform best in each state. Also, as H-Agent does not interact directly with the environment, the states that describes the traffic light, E_{ft}, are irrelevant and, therefore, disregarded. Thus, H-Agent describes the intersection through the combination of the 9 E_q states and the 9 E_w states, totalling 81 states.

Similarly to the sub-policies, H-Agent also has a single set of actions available, O, composed of the two *options*, making its table $Q(s, o)$ to have two columns, totalling 162 states. The reward r, in turn, was defined as the relationship between the number of vehicles served (V_s) and the average waiting time (V_w), as in (5). Both values are normalized, using Min-max [37], in the range [0, 1]. The lower limit of flow (V'_f) is 0. Its upper limit, which represents the maximum size of a queue, is 50. The lower limit of waiting time (V'_w) is 10, which is the minimum green time. Its upper limit is 30, which is the cycle time according to the green and red-amber times set for the HRL agent. The constant 0.0001 ensures that no division by zero occurs.

$$r = \frac{V'_f}{V'_w + 0.0001} \tag{5}$$

The hyperparameters values determined after an exhaustive search were $\alpha = 0.2$ and $\gamma = 0.9$. As H-Agent selects between two sub-policies, both must also undergo a training process. In order to be able to analyze only the final performance presented by H-Agent, the two sub-policies were trained before the test rounds, with the hyperparameter values previously determined. At the top level of the hierarchy, there is still a third parameter to be configured, the sub-policy stopping criterion (β), which can be configured to interrupt the sub-policy in three ways: 1) when a maximum number of primitive actions is achieved; 2) when a specific objective (state) is achieved; or 3) when it becomes more advantageous to assume a sub-policy different from the current one. Considering that $\beta = 1$ indicates that the sub-policy must be stopped, (6) summarizes its behavior.

$$\beta_O = \begin{cases} 1 & \text{if the number of steps} > \text{k} \\ 1 & \text{if } Q(s_{t+1}, o') > Q(s_{t+1}, o) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The number of steps taken (k) is the only configurable value, which may be specific to each sub-policy. To choose its values (k_q and k_w), the same set of definition tests from α and γ were applied to k (0.2 and 0.9, respectively - optimal). The set of possible values considered was [2,3,4,5] for each sub-policy, generating a universe of 16 possible pairs. These pairs were submitted to the first round of tests and, in the end, $k_q = 5$ and $k_w = 2$ were empirically obtained. There is no way to run an H-Agent training without defining α, γ and k. Ideally, all hyperparameters would be explored and determined together, but to shorten the already extensive training phase, we chose to split this phase in two: The first was to determine α and γ, with a fixed $k = 5$; and the second to determine only k, using the optimal pair obtained before, going from a universe of $10 \cdot 9 \cdot 4 \cdot 4 = 1440$ training scenarios to a total of $10 \cdot 9 + 4 \cdot 4 = 106$.

After the hyperparameters tuning phase, the H-Agent was subjected to the test scenarios, with its results taken from the average of 10 training/testing cycles. Table 3 shows the results. The best ones are highlighted in bold and the worst in italics. As well as the choice of hyperparameters, the sub-policies were trained separately. Therefore, the results shown in the table are the result of repeated tests in the macro policy of H-Agent only.

TABLE 3. Summarized results from H-Agent, Q-Size, W-Size and FT approaches.

Scenario	V_s	V_m	W
H-Agent			
Fixed-1800	1781	29.68	19.08
Peak-2700	2428	40.47	21.01
Peak-3600	2923	48.73	**22.8**
W-Size			
Fixed-1800	1782	*29.23*	17.14
Peak-2700	2396	39.93	**19.05**
Peak-3600	2826	47.03	23.21
Q-Size			
Fixed-1800	*1779*	29.66	*25.09*
Peak-2700	**2433**	**40.56**	25.32
Peak-3600	**2929**	**48.83**	27.79
Fixed-Time			
Fixed-1800	**1784**	**29.73**	**16.63**
Peak-2700	*2349*	*39.15*	*29.34*
Peak-3600	*2420*	*40.33*	*31.53*

Again, the values of total flow and flow per minute observed for Fixed-1800 are similar for all approaches. Regarding the average waiting time in this scenario, the H-Agent has a performance between Q-Size and W-Size, which is 23.9% more efficient than the first and 11.3% less efficient than the second. From the Peak-2700 scenario, it becomes clearer that the H-Agent results are a mix between the two previous sub-policies, in which its flow rate is virtually equal to that of Q-Size (2428 and 2433, respectively), and its waiting time is close to that of W-Size, with a difference of 10.28%. Figures 9 and 10 corroborates this behavior of the H-Agent. They also show that the H-Agent flow is similar to that of Q-Size, and its waiting time is closer to that of W-Size, with variations according to demand, but always closer to the average. This indicates stability.

The results of Peak-3600, illustrated in Figures 11 and 12, clearly show the ability of H-Agent to combine the performances of Q-Size and W-Size. Its result for total flow is similar to that of Q-Size (2923 and 2929), but reaches higher

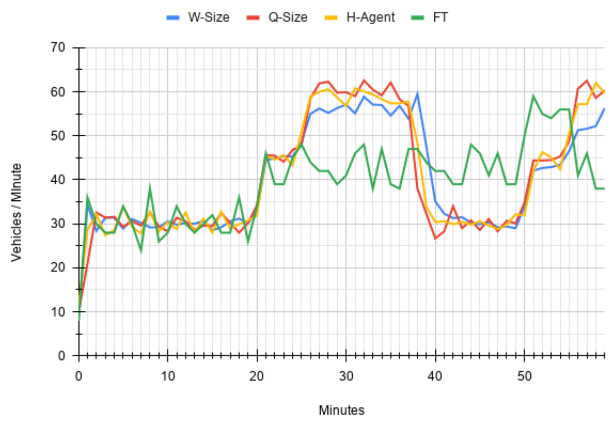


FIGURE 9. Flow results of H-Agent, W-Size and Q-Size in Peak-2700.

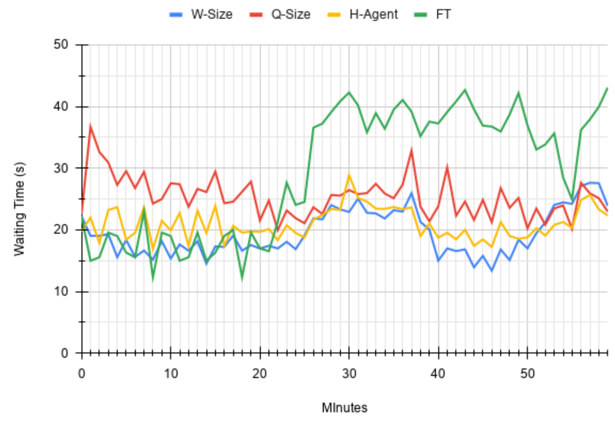


FIGURE 10. Average waiting time results of H-Agent, W-Size and Q-Size in Peak-2700.

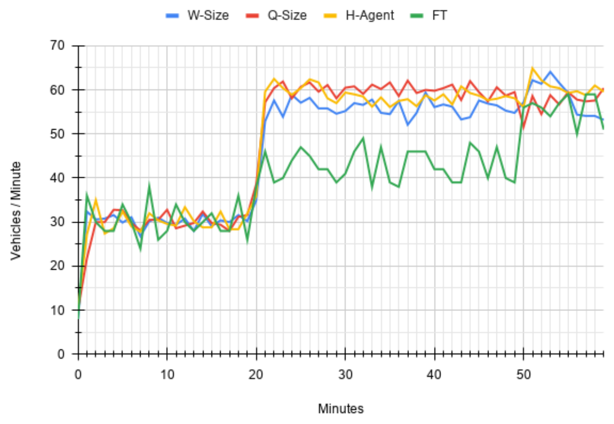


FIGURE 11. Flow results of H-Agent, W-Size and Q-Size in Peak-3600.

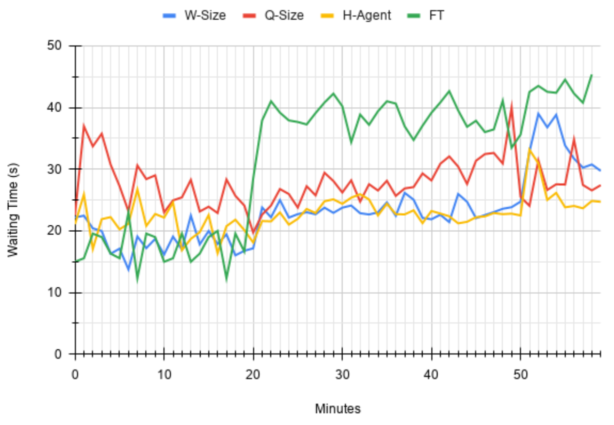


FIGURE 12. Average waiting time results of H-Agent, W-Size and Q-Size in Peak-3600.

values for longer. For example, between minutes 21-30 or from 50 minutes onward, the H-Agent has the highest and most stable flow of all approaches. Moreover, the H-Agent not only has a waiting time close to W-Size, but its result is 1.76% smaller, with less frequent performance loss peaks (minute 52). It is worth noticing that H-Agent managed to obtain results similar to those from each individual sub-policy, without sacrificing what both renounced. As an evidence of its potential, not a single line of the H-Agent section of Table 3 was marked in italics, which highlights the worst results. In fact, although only one item was marked as best (waiting time of Peak-3600), all values are very close to those marked as best (bold) in other sections (Q-Size, W-Size, FT). To sum up, H-Agent reaches the most balanced results, close to the best and far from the worst.

The final comparison is made between the Fixed-Time approach (FT) and the H-Agent based on HRL. Although the FT model seems to perform well enough for well-behaved scenarios, such as Fixed-1800, it is unable to consider traffic variation during the day and to handle unexpected traffic bursts. It is shown in Figure 9 (Peak-2700) how the HRL agent, even with shorter green times (10 seconds vs. 15 seconds), can adapt itself to the growing traffic demand. Despite the difference between green times, the H-Agent is able to select the same lane over and over again, which, in fact, changes the lane's green time in increments of 10 seconds. This characteristic makes the system more capable. However, it is worth questioning that, if the agent can choose a single lane over and over again, then the waiting time of the opposite lane will rise up. In fact, this is what is observed when using only Q-Size as a policy. Q-Size is able to increase the flow, but negatively impacts the waiting time. However, as the H-Agent can also assume the behavior of W-Size, it manages to overcome the deficiency of Q-Size. Figure 10 shows that, even with a flow greater than FT, H-Agent keeps the waiting time stable and lower. Very similar and more evident conclusions can be obtained from the analysis of the results in Peak-3600 (Figures 11 and 12).

The values presented by the H-Agent based on HRL show that it is possible to combine two sub-policies and generate results superior to their isolated performances, even in their field of specialty. This is due to the fact that the HRL agent, guided by its reward model, seeks to initiate its sub-policies at times (states) when maximum performance can be obtained by them. In this way, the disadvantage that each sub-policy presents, due to having only one way of interacting with the environment, is not only mitigated, but also supplanted by the capabilities of each other, ultimately creating a more stable and capable agent. For this reason, the H-Agent can be considered the best choice, as it combines the best features from its sub-policies and overcomes the greatest issue regarding the FT model: its inability to adapt to the ever-changing environment.

V. CONCLUSION

Webster Method is the most common approach for calculating phase times for traffic lights. Its formula is based on immutable values such as lane width, maximum speed, and maximum/peak flow of vehicles at the intersection. Nevertheless, the flow of vehicles is variable and can be affected by random events, which can lead to values very different from those initially considered.

The use of Artificial Intelligence (AI) models, able to make decisions from environment data in a dynamic way, presents itself as a viable solution to this problem. In particular, Reinforcement Learning (RL) is an AI technique which has capabilities that fit the observed problem: *online* learning model and *model-free* approach. In other words, an RL agent continues to learn while in operation, and does not need to know the complete process to infer its behavior and make decisions. This study aims to apply Hierarchical Reinforcement Learning (HRL) and Options Framework to traffic light control and compare its performance to that of a fixed-time traffic controller (FT), which uses the Webster Method. All test scenarios were built using the SUMO simulation tool.

Two objectives were established for the HRL agent, each forming a sub-policy: maximizing vehicle flow (Q-Size) and minimizing waiting times (W-Size). The determination of which sub-objective to pursue is made by the main policy (H-Agent), which selects one of the sub-policies at the time when maximum performance can be obtained by it. Also, each sub-policy is a complete *Q-learning* process that could be used individually. It is possible not only to compare the results obtained by HRL and FT models, but also with agents that use Q-Size or W-Size sub-policies exclusively.

The results demonstrate that it is possible to combine two sub-policies (Q-Size and W-Size) and generate results superior to their isolated performances, even in their field of specialty. In this way, the disadvantage that each sub-policy presents when applied individually is supplanted by the capabilities of the other, creating a more stable hybrid agent (H-Agent). In relation to the FT model, all intelligent agents present significant improvements, between 20% and 30%, with a higher number of vehicles served and lower waiting times in line, especially for scenarios with unexpected traffic growth. Finally, the H-Agent can be considered the best choice, as it combines the core features from its sub-policies and presents more balanced results. Although the FT model performs consistently for well-behaved scenarios, it is still unable to consider traffic variation during the day and to handle unexpected traffic bursts.

In the future, it is planned to apply the proposed hierarchical agent to more complex scenarios, mainly with more intersections, to analyze if there is any impact on its performance. It is also possible to apply a Deep Learning approach, with a model that uses a Neural Network to approximate the Q-function, allowing a more complex and rich set of inputs/features and more complex and assertive actions.

REFERENCES

[1] Denatran. (2020). *Frota de Veiculos*. [Online]. Available: http://www.denatran.gov.br/estatistica/237-frota-veiculos

[2] E. E. Agency. (2020). *Size of the Vehicle Fleet in Europe*. [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/size-of-the-vehicle-%fleet/size-of-the-vehicle-fleet-10

[3] *Manual de Semaforos*, Denatran, Departamento Nacional de Transito, Brasilia, Brazil, 1984.

[4] F. V. Webster and B. M. Cobbe, *Traffic Signals*. Her Majesty's Stationary Office, Richmond, U.K., 1966.

[5] N. Vandaele, T. V. Woensel, and A. Verbruggen, "A queueing based traffic flow model," *Transp. Res. D, Transp. Environ.*, vol. 5, no. 2, pp. 121–135, Mar. 2000.

[6] R. Jerry Okecukwu Ekeocha and V. Ikechi Ihebom, "The use of queuing theory in the management of traffic intensity," *Int. J. Sci.*, vol. 4, no. 3, pp. 56–63, 2018.

[7] A. D. Febbraro, D. Giglio, and N. Sacco, "Urban traffic control structure based on hybrid Petri nets," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 224–237, Dec. 2004.

[8] T. Riedel and U. Brunner, "Traffic control using graph theory," *Control Eng. Pract.*, vol. 2, no. 3, pp. 397–404, Jun. 1994.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] D. Zhao, Y. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 485–494, Jul. 2012.

[11] M. Bielli, G. Ambrosino, M. Boero, and M. Mastretta, "Artificial intelligence techniques for urban traffic control," *Transp. Res. A, Gen.*, vol. 25, no. 5, pp. 319–325, Sep. 1991.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[13] R. Aragon-Gómez and J. B. Clempner, "Traffic-signal control reinforcement learning approach for continuous-time Markov games," *Eng. Appl. Artif. Intell.*, vol. 89, Mar. 2020, Art. no. 103415. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197619303239

[14] M. A. Khamis and W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Eng. Appl. Artif. Intell.*, vol. 29, pp. 134–151, Mar. 2014.

[15] P. K. J., H. Kumar A. N, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 2529–2534.

[16] B. Hengst, C. Sammut, and G. I. Webb, *Hierarchical Reinforcement Learning*. Boston, MA, USA: Springer, 2010, pp. 495–502.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[18] D. F. Borges, E. M. Moreira, A. D. de Souza, and J. P. R. Leite, "Traffic light control and machine learning: A systematic mapping review," in *Proc. ITNG 18th Int. Conf. Inf. Technol.-New Gener.* Cham, Switzerland: Springer, 2021, pp. 11–17.

[19] M. Kubat, *An Introduction to Machine Learning*. New York, NY, USA: Springer, 2017.

[20] 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 Comput. Surv.*, vol. 50, no. 3, pp. 1–38, Oct. 2017.

[21] A. Markov, "Extension of the limit theorems of probability theory to a sum of variables connected in a chain," in *Dynamic Probabilistic Systems (Markov Models)*, vol. 1, R. Howard, Ed. New York, NY, USA: Wiley, 1907, ch. reprinted in Appendix B, pp. 552–577.

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[23] A. G. Barto and S. Mahadevan, "Recent advances on hierarchical reinforcement learning," *Discrete Event Dyn. Syst.*, vol. 13, no. 4, pp. 341–379, 2003.

[24] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. Burlington, MA, USA: Morgan-Kaufmann, 1993, pp. 271–278.

[25] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, 2000.

[26] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, 1999.

[27] M. C. Machado, M. G. Bellemare, and M. Bowling, "A Laplacian framework for option discovery in reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 2295–2304.

[28] A. Solway, C. Diuk, N. Córdova, D. Yee, A. G. Barto, Y. Niv, and M. M. Botvinick, "Optimal behavioral hierarchy," *PLoS Comput. Biol.*, vol. 10, no. 8, Aug. 2014, Art. no. e1003779.

[29] P. A. Lopez, E. Wiessner, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, and P. Wagner, "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[30] SUMO. (Sep. 2020). *Simulation of Urban Mobility*. [Online]. Available: https://www.eclipse.org/sumo/

[31] J. Zeng, J. Hu, and Y. Zhang, "Adaptive traffic signal control with deep recurrent Q-learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1215–1220.

[32] G. B. Castro, A. R. Hirakawa, and J. S. C. Martini, "Adaptive traffic signal control based on bio-neural network," *Procedia Comput. Sci.*, vol. 109, pp. 1182–1187, Jan. 2017.

[33] J. Jin and X. Ma, "Adaptive group-based signal control by reinforcement learning," *Transp. Res. Procedia*, vol. 10, pp. 207–216, Jan. 2015.

[34] *Traffic Control Interface*, SUMO, Redwood City, CA, USA, Sep. 2020.

[35] R. Larson, *Elementary Statistics : Picturing the World*. Boston, MA, USA: Pearson, 2015.

[36] M. Lapan, *Deep Reinforcement Learning Hands-on*. Birmingham, U.K.: Packt Publishing, 2018.

[37] C. C. Aggarwal, *Data Mining*. Cham, Switzerland: Springer, 2015.

DIMITRIUS F. BORGES received the bachelor's degree in computer engineering from the National Institute of Telecommunications (INATEL), Brazil, in 2012, and the M.Sc. degree in computer science and technology from the Federal University of Itajubá (UNIFEI), Brazil. From 2012 to 2019, he worked in the development of hardware and software used in vehicular traffic control, in his family company, SEMA-SEG, later leaving it to start his current job, working on the development of tracking and telemetry systems for intelligent transport systems at Wplex Software Company.

JOÃO PAULO R. R. LEITE received the B.Sc. degree in computer engineering from the Federal University of Itajubá (UNIFEI), Brazil, in 2007, and the M.Sc. degree in computer science and technology and the Ph.D. degree in electrical engineering from UNIFEI, in 2010 and 2018, respectively. He is currently an Adjunct Professor at UNIFEI, with focus on artificial neural networks and artificial intelligence, analysis of algorithms, data structures, and digital signal processing. He is a member of the Research Group on Systems and Computer Engineering, UNIFEI.

EDMILSON M. MOREIRA received the B.Sc. degree in computer science from the University of Alfenas, Brazil, in 1994, the B.Sc. degree in mathematics from the Faculty of Philosophy, Sciences and Linguistics, Varginha, Brazil, in 1996, and the M.Sc. and Ph.D. degrees in computer science and computational mathematics from the University of São Paulo (USP), Brazil, in 2000 and 2005, respectively. He is currently an Associate Professor with the Federal University of Itajubá, Brazil. He works with graph theory, distributed systems, and discrete events simulation. He is a member of the Research Group on Systems and Computer Engineering, Federal University of Itajubá.

OTÁVIO A. S. CARPINTEIRO was born in Rio de Janeiro, Brazil. He received the B.Sc. degree in mathematics, the B.Sc. degree in music, the M.Sc. degree in systems and computer engineering from the Federal University of Rio de Janeiro, Rio de Janeiro, and the D.Phil. degree in cognitive and computer science from the University of Sussex, U.K. For many years, he has worked as a System Analyst and ended his professional career as a Full Professor with the Federal University of Itajubá, Minas Gerais, Brazil, where he did research, supervised graduate students, and taught undergraduate and graduate courses in computer engineering. He is currently retired, but is still working as a member of the Research and Development Projects, Research Group on Systems and Computer Engineering, Federal University of Itajubá.