A Methodology for the Development of RL-Based Adaptive Traffic Signal Controllers

Guilherme S. Varela,[1] **Pedro P. Santos,**[1] **Alberto Sardinha,**[1] **Francisco S. Melo**[1]
[1] INESC-ID and Instituto Superior Técnico
University of Lisbon, Portugal
guilherme.varela@tecnico.ulisboa.pt, pedro.pinto.santos@tecnico.ulisboa.pt,
jose.alberto.sardinha@tecnico.ulisboa.pt, fmelo@inesc-id.pt

Abstract

This article proposes a methodology for the development of adaptive traffic signal controllers using reinforcement learning. Our methodology addresses the lack of standardization in the literature that renders the comparison of approaches in different works meaningless, due to differences in metrics, environments and even experimental design and methodology. The proposed methodology thus comprises all the steps necessary to develop, deploy and evaluate an adaptive traffic signal controller—from simulation setup to problem formulation and experimental design. We illustrate the proposed methodology in two simple scenarios, highlighting how its different steps address limitations found in the current literature.

Introduction

Traffic congestion is a cross-continental problem. In the United States alone, an average automobile commuter spends 54 hours in congested traffic and wastes 21 gallons of fuel due to congestion, leading to a total estimated cost of $1,080$ USD in wasted time and fuel per commuter (Schrank, Eisele, and Lomax 2019), not considering external costs such as the increasing price of goods caused by the inflation of transportation costs, environmental and productivity impacts, as well as the decrease of population's quality of life (Hilbrecht, Smale, and Mock 2014). Similarly, a recent study shows that, in the EU, the total external costs associated with traffic is over $300,000$ million euros (Becker, Becker, and Gerlach 2016). Hence, there have been numerous initiatives to mitigate traffic congestion, such as investment in public transit systems (Harford 2006) or intelligent transportation systems (Dimitrakopoulos and Demestichas 2010).

Traffic signals, being a fundamental element in traffic control and regulation, are at the same time responsible for a significant percentage of traffic bottlenecks in urban environments, and play a key role in addressing the problem of traffic congestion. Effective traffic signal control is, therefore, a key part of urban traffic management. Classic traffic signal control approaches from transportation engineering, such as the Webster (Webster 1958) or Max-Pressure (Varaiya 2013) methods, are capable of greatly increasing the efficiency of traffic infrastructures all around the world. However, such approaches are either unable to adapt to changing traffic volumes, or rely on oversimplified traffic models, manual-tuning and inaccurate traffic information (Zheng et al. 2019; Wei et al. 2019).

Alternatively, *adaptive traffic signal control* (ATSC) approaches seek to take advantage of the multiple sources of information currently available (from mobile navigation applications, ride sharing platforms, etc.). Machine learning techniques can use the data made available by such platforms to provide traffic signal control strategies that adapt to the current traffic conditions in an effective manner, providing a promising alternative to classical approaches.

Recently, several researchers have addressed ATSC using Markov decision processes (MDP) (Wang et al. 2019; Wei et al. 2019). MDPs model discrete-time stochastic control problems, and are extensively used in artificial intelligence to describe problems of sequential decision-making under uncertainty (Puterman 2005). In an MDP, an agent (the controller) interacts sequentially with an environment by selecting actions based on its observation of the environment's state; the actions selected by the agent influence how the environment's state evolves, and the agent receives a numerical evaluation signal (a reward) that instantaneously assesses the quality of the agent's action. The agent's goal is to select its actions to maximize some form of cumulative reward. MDPs are the backbone of *reinforcement learning* (RL), a machine learning paradigm in which the agent learns the optimal way of selecting the actions from direct interaction with the environment, without resorting to any pre-defined model thereof (Sutton and Barto 1998).

RL algorithms are a natural choice when addressing ATSC, since they can be trained directly in the data available, without requiring human annotators to define what is a "good" or "bad" control strategy. Unfortunately, the use of RL in this domain is not without its own challenges.

One of the first challenges is the lack of standard environments in the domain of ATSC. The RL field has benefited from standardized environments and easy to use APIs that allow researchers to compare different approaches to the same problem space, e.g the Deep Q-network (Mnih et al. 2013) which has been shown to generate relevant features for the Arcade Learning Environment. The need for standardization has sparkled new research towards open source frameworks (Genders and Razavi 2019), as means to prevent researchers to re-implement the same set of fundamental tools with which to conduct *de facto* experiments. We argue it is important that the community agrees on a set of benchmark environments/traffic networks that may be used as a first test stage for the algorithms explored in the context of ATSC. The existence of such benchmarks would enable a proper comparison of different models and algorithms in a common set of environments, enabling a clearer assessment of the strengths and weaknesses of different alternatives.

Another major challenge is related to the *security* and *explainability* of current RL architectures. Although some RL systems have been quite successful in improving metrics of interest such as the average travel time, some protocols are not viable to be implemented in a real world situations for a variety of reasons (Ault, Hanna, and Sharon 2020) such as, long and unsafe tuning process, opaque policies, and the predominant use of synthetic simulation scenarios.

One third challenge is *reproducibility*. Reproducibility is thoroughly documented in RL literature, as such systems might overfit to the training experience, showing good performance during training but performing poorly at deployment time (Whiteson et al. 2011). Works show that simply changing the random seeds used to generate the simulations influence in a statistically significant manner the outcome of the RL algorithm (Aslani, Mesgari, and Wiering 2017).

This paper contributes one further step towards a wider application of RL in ATSC. However, for this potential to be realized, it is paramount to address the standardization, security and reproducibility issues identified above. Our contribution in this article is, therefore, a methodology for the development of RL-based adaptive traffic signal controllers that ensures some level of standardization at the different stages of the experimental process: simulation setup, environment modeling, experimental design and result reporting. We adopt an action definition which is rather constrained: it produces synchronous joint action schemes across the network – we show that the reinforcement learning agent is able to find policies that are on par with classical controllers which benefit from both human supervision and from decades-old literature from the Transportation domain. Such action plans generate protocols which are more in line with governmental transportation department's expectations, hence they provide more trust in face of the liability that such regulatory agencies face. In particular, with respect to experimental design and result reporting, we discuss good practices and relevant metrics that have been explored in different works, highlighting their merits and how their adoption may contribute to better interpretation of experimental results and mitigate reproducibility issues. We illustrate our own methodology by applying its different steps in designing a traffic signal controller using the well known deep Q-network (DQN) algorithm (Mnih et al. 2013).

Background

Most approaches to traffic signal control rely on computer software for *microsimulation*, which simulates traffic at the level of individual vehicles, computing the position, velocity, emissions data and other for every vehicle at each time step. A *route* is a sequence of roads used by vehicles to traverse the network.

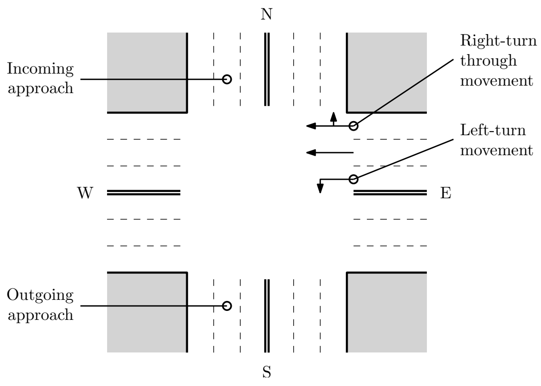

Figure 1: Intersection with four incoming approaches, each composed of three lanes.

A traffic infrastructure can be represented as a network/graph, where *roads* and *junctions* correspond to the edges and nodes, respectively. A road connects two nodes of the network and has a given number of *lanes*, a maximum speed and a length. Traffic light controllers are typically installed at road junctions. An intersection is a common type of junction in which roads cross each other. Figure 1 illustrates a typical intersection with four incoming and four outgoing *approaches*, each approach composed of three lanes.

A *signal movement* refers to the transit of vehicles from an incoming approach to an outgoing approach. A signal movement can generally be sub-categorized as a *left-turn*, *through* or *right-turn* movement. For example, in the intersection of Fig. 1, East-South corresponds to a left-turn movement, while East-West corresponds to a through movement. Both are examples of signal movements. A green signal indicates that the respective movement is allowed, whereas a red signal indicates that the movement is prohibited.

A *signal phase* is a combination of non-conflicting signal movements, i.e. the signal movements that can be set to green at the same time. In the intersection of Fig. 1, the triplet (North through, South through, South right-turn) is a valid signal phase. In contrast, (North through, South left-turn) is not. A *signal plan* for a single intersection is a sequence of signal phases and their respective durations. Usually, the time to cycle through all phases, known as *cycle length*, is fixed. Therefore, the phase splits—i.e., the amount of time allocated for each signal phase—are normally defined as a ratio of the cycle length. A yellow signal is set as a transition from a green to a red signal.

Reinforcement Learning

As mentioned in the introduction, reinforcement learning considers an agent whose interaction with the environment can be described as a *Markov decision process* (MDP). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \{\mathcal{P}_a, a \in \mathcal{A}\}, r, \gamma)$, where $\mathcal{S}$ is a set of *states* and $\mathcal{A}$ is a set of *actions*. At each step t, the agent observes the state $S_t \in \mathcal{S}$ of the environment and selects an action $A_t \in \mathcal{A}$. The environment then transitions to a state

S_{t+1}, where

$$\mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right] = [\mathbf{P}_a]_{ss'}. \quad (1)$$

The matrix $\mathbf{P}_a, a \in \mathcal{A}$, encodes the *transition probabilities* associated with action a. Upon executing an action a in state s, the agent receives a (possibly random) reward with expectation given by $r(s, a)$. The goal of the agent is to select its actions so as to maximize the *expected total discounted reward* (TDR),

$$\mathrm{TDR} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \quad (2)$$

where R_t is the random reward received at time step t (with $\mathbb{E}[R_t] = r(S_t, A_t)$) and the scalar γ is a *discount factor*. The long-term *value* of an action a in a state s is captured by the optimal Q-value, $Q^*(s, a)$, which can be computed using, for example, the *Q-learning algorithm* (Watkins 1989). The Q-learning algorithm estimates the optimal Q-values as the agent interacts with the environment: given a transition (s, a, r, s') experienced by the agent, Q-learning performs the update

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha\big(r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a') - \hat{Q}(s,a)\big), \quad (3)$$

where α is a step size. Upon computing Q^*, the agent can act optimally by selecting, in each state s, the optimal action at s, given by $\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$. The mapping $\pi^* : \mathcal{S} \to \mathcal{A}$, mapping each state s to the corresponding optimal action $\pi^*(s)$, is known as the *optimal policy* for the MDP.

Deep Q-network (Mnih et al. 2013) is a well known RL method that approximates the Q-values with a neural network $\hat{Q}(s, a; \theta)$, where θ denotes the parameters of the model. At each step, the agent adds a transition (s, a, r, s') to a replay memory buffer, from which batches of transitions are sampled in order to optimize the parameters of the model such that the following loss in minimized:

$$\mathcal{L}(\theta) = \left(r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'; \theta^-) - \hat{Q}(s, a; \theta)\right)^2. \quad (4)$$

The gradient of the loss is backpropagated only into the *behaviour network*, $\hat{Q}(s, a; \theta)$, which is used to select actions. The term θ^- represents the parameters of the *target network*, a periodic copy of the behaviour network.

Related Work

Several works have explored the use of RL in traffic light control, most of which rely on estimating the Q-function or an approximation thereof (Abdulhai, Pringle, and Karakoulas 2003; Wiering 2000). In their simplest form, such RL approaches consider that each intersection is controlled by a single agent that ignores the existence of other agents in neighboring intersections. More sophisticated approaches consider the existence of multiple agents and leverage the network structure to address the interaction between the different agents (Prabuchandran, Hemanth, and Bhatnagar 2014; Liu, Liu, and Chen 2017). With the advent of deep learning, several of the approaches above have been extended to accommodate deep neural networks as the underlying representation of the problem. For example, Genders and Razavi (Genders and Razavi 2016) propose a DQN control agent that combines a deep convolutional networks with Q-learning. The work controls the traffic lights at a single intersection, and essentially extends previous work to accommodate the deep learning model.

In terms of experimental methodology, there are several issues that make the comparison of different approaches challenging. First, different works adopt different evaluation metrics and baseline policies. In one work, the performance metrics used are the average travel time per car and the average wait time per car; the proposed approach is compared against a fixed timed plan (Thorpe 1997). In a different work, the metrics adopted are the average waiting time and number of refused cars (a saturation condition of the used simulator), and the proposed approach is compared against both a random policy and a fixed-time policy (Wiering 2000). In yet another work, the metrics used are the ratio of the average delay, and the proposed approach is compared against a pre-timed plan (Abdulhai, Pringle, and Karakoulas 2003). The lack of a clear understanding of the dependence of the different metrics on the number of intersections is another issue that renders comparisons difficult. Finally, while average metrics of the variables of interest are provided, most works offer no measures of significance regarding the reported performance.

Recently, several works sought to address some of the issues previously discussed (Genders and Razavi 2019). Researchers have put forth several recommendations/good practices when exploring the use of RL in the context of ATSC: (i) provide all hyper-parameters and number of trial experiments; (ii) report aggregated results with deviation metrics (averages and standard deviations, not maximum returns); (iii) implement proper experimental procedures (average together many trials using different random seeds for each) (Islam et al. 2017). We extend those works by providing both the preliminary steps necessary to simulate, develop, train and evaluate RL-based experiments for ATSC on real world scenarios, as well as, insights which can be extracted by our methodology in this domain.

Methodology

We propose a four-stage methodology to be used in the development of RL-based traffic signal controllers. Figure 2 illustrates the proposed methodology, comprising four phases: simulation setup, MDP formulation and selection of the RL learning method, train and evaluation.

Simulation setup

The first stage of the proposed methodology is the *simulation setup* phase. RL controllers must be trained by resorting to (micro-)simulators that are able to provide a realistic response to the agent's actions during the learning process. The main objective of the simulation setup stage is to prepare all the simulations needed to carry out such training. It includes gathering simulation-related data such

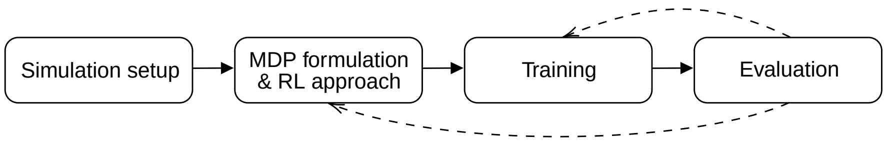

Figure 2: Diagram illustrating the proposed methodology, composed of four stages. Solid arrows denote the main development flow, whereas dashed arrows denote the iterative process of model tuning.

as the topological data of the roads' network and vehicles demand/routes data, as well as setting up the traffic simulator.

In any case, for the purpose of our methodology, it is important that during this stage two key components are configured and defined: (i) the topology of the roads network; and (ii) the traffic demands and routes.

Roads network topology. Networks can be either synthetic or extracted from real world locations. Available open source services, such as OPENSTREETMAP (OpenStreetMap contributors 2017), allow segments of cities' districts to be exported and, during the simulation setup step, such information can be prepared and fed to the simulator, thus opening up the possibility of simulating a rich set of networks relevant to real-world traffic signal control.

In our own implementation, we use geospatial data from OPENSTREEMAPS to build the configuration files to be used by the simulator, in our case, the SUMO micro-simulator (Krajzewicz et al. 2012). The following steps are required: (i) extract the region of interest from OPENSTREETMAP and open the resulting file with the JOSM[1] editor, an extensible editor for OPENSTREETMAP files, in order to fine-tune the network; (ii) convert the (edited) OPENSTREETMAP file into the SUMO network format using the `netconvert`[2] tool; and (iii) open the resulting SUMO network file with the `netedit`[3] tool, a graphical network editor for SUMO, in order to ensure that all intersections are properly setup, namely check whether all traffic phases and links (connections between lanes) are correct.

Traffic demands and routes. Traffic demands and routes can be either synthetic (Wei et al. 2018) or derived from real-world data using origin-destination matrices (Aslani, Mesgari, and Wiering 2017) or induction loops counts (Rodrigues and Azevedo 2019). Regarding synthetic demands, simple (constant demands) to complex (variable demands) scenarios can be created by specifying the probabilites of vehicles' insertion through time. With respect to the generation of a synthetic set of routes, the `duarouter`[4] tool can be used. Afterwards, a probability can be assigned to each unique route by weighting it accordingly to a pre-determined criteria, such as, invertionally proportional to the number of turns contained in the respective route.

[1]https://josm.openstreetmap.de/
[2]https://sumo.dlr.de/docs/netconvert.html
[3]https://sumo.dlr.de/docs/netedit.html
[4]https://sumo.dlr.de/docs/duarouter.html

MDP formulation and RL approach

The second stage in our methodology is the description of the traffic control problem as a Markov decision problem (or a multiagent version thereof). As previously discussed, an MDP comprises 5 elements: the set of *states*, the set of *actions*, the *transition probabilities*, the *reward*, and the *discount factor*.

Most works which apply RL to the ATSC domain do not specify the transition probabilities since they are not strictly required – additionally explicitly modeling the traffic dynamics as a result of changes in traffic light control is unfeasible in most cases. The remaining four components must still be specified: the state space (i.e., the information upon which the agent will base its decisions), the action space (i.e., how the agent is able to influence the environment through the choice of its actions) and the reward function (which implicitly encodes the goal of the agent). The literature is very diverse with respect to the adopted problem formulation; it is important to stress that the performance of the resulting controller will depend critically on the choices made at this stage.

Alongside the MDP formulation, it is also necessary to select the desired RL method to be used as a learning component for the traffic signal controller, a choice that is not independent of MDP formulation. The literature is also very diverse with respect to the algorithm choice, even though the majority of the works focus their attention on the study of value-based methods (i.e., methods that seek to estimate/approximate the Q-function or a surrogate thereof).

The work of El-Tantawy et al. (El-Tantawy, Abdulhai, and Abdelgawad 2014) provides a comparison between some common state-space representations, reward functions and action space definitions, using a real-world network topology. Wei et al. (Wei et al. 2019) provide a comprehensive list of commonly used MDP formulations in the context of ATSC, as well as RL methods used in the context of traffic signal control.

Training

The *training* procedure of RL-based traffic signal controllers should follow the same guidelines used in the field of RL. For example, a proper balance between exploration and exploitation should be ensured, making sure that the agent is able to experience a wide range of different situations. In adherence to the good practices previously discussed, it is important to run multiple instances of the training process, using different seeds, in order to correctly assess the learning ability of the proposed RL method.

In the context of ATSC, particular attention must be paid to ensure that the simulations are properly running. *Gridlocks* should be avoided or properly processed, for example by restarting the simulation, adjusting the vehicles' arrivals, or teleporting vehicles.[5]

Finally, it is important to monitor performance metrics such as losses, rewards, the number of vehicles in the simulation and the vehicles' velocity throughout the training in order to gain a better insight into the learning process.

The outcome of the training process consists of a set of policies (each one resulting from a different training run), that need to be properly evaluated. The next and final step of the proposed methodology addresses how this can be accomplished.

Evaluation

In the context of traffic light control, several performance metrics have been proposed: travel time, waiting time, number of stops, queue length, throughput, as well as gas emissions and fuel consumption. From all these metrics, the minimization of the travel time is usually the main goal in the development of ATSCs, therefore, it is arguably the most important metric to report. Since algorithms are usually unable to directly optimize the travel time, it is useful to report additional metrics that are more closely related with the formulated agent's objective (the reward function, in the case of RL agents), such as the queue length or waiting time.

It is important to run some baseline algorithms, such as the Webster or Max-Pressure methods, under the same scenario. These runs are of extreme importance since they allow to compare the performance of the RL controller(s) against well-established, commonly used traffic engineering methods.

Performance estimation. In order to adequately compare the different approaches, it is important to assess the performance of the alternative proposals with a set of accordingly seeded simulations in order to rule out any influence of the simulation seeds in the results. For the baseline algorithms, this can be achieved by simply running multiple evaluation simulations. With respect to the RL agents, each of the policies that resulted from the training stage should be evaluated with a set of evaluation rollouts and, if posteriorly needed, the results aggregated per policy.

Performance analysis & comparison. The simplest and most straightforward way to present and compare the performances of the different methods is through point estimation, for example by reporting the mean and standard deviation of the travel times observed for a set of evaluation rollouts. While this is commonly used in the ATSC domain, sometimes it exists a big overlap in the reported performance metrics between different methods, thus, it is important to understand whether the observed differences are statistically significant or not.

[5]A gridlock occurs when a queue from one bottleneck creates a new bottleneck somewhere else, and so on in a vicious cycle that completely stalls the vehicles' circulation (Daganzo 2007).

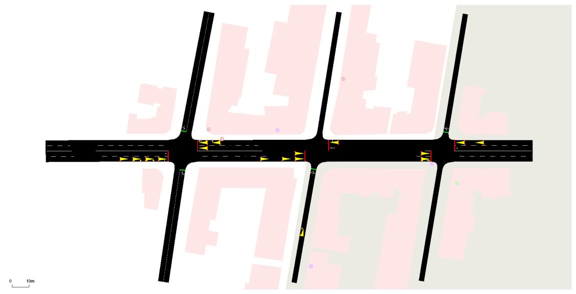

Figure 3: SUMO screenshot of the considered intersections, near Marquês de Pombal square, in Lisbon. Scenario 1 comprises only the left-most intersection whereas scenario 2 is composed of all three pictured intersections. Phase 1 allows the movement of vehicles in the vertical direction whereas phase 2 allows the movement in the horizontal direction.

A better insight into the performance of the methods can be achieved by plotting the distributions of the travel time means for each of the methods, however, in order to draw conclusions from the observed performances it is proposed the use of statistical hypothesis testing (Ross 2004). Specifically, we are interested in trying to understand whether two or more population means (the performance metrics of the different methods) are equal. It is highly likely that the previously hypothesis is rejected, therefore, *post hoc* comparisons need to be performed in order to understand between which mean pairs exists a statistically significant difference. In order to perform such evaluation, it is proposed the use of the (one-way) ANOVA test, followed by the Tukey Honestly Significant Difference (HSD) test for *post hoc* comparisons. Unfortunately, the previous tests make some assumptions related with the data distributions that not always hold. Therefore, it is worth checking whether the assumptions are met, and if not, to switch the aforementioned tests with their respective non-parametric versions.

As a complementary analysis, it might be interesting to plot histograms for the performance metrics as most of the times, mean values may not be well representative of the underlying distributions.

Experiments

We now illustrate the application of the previously described methodology by developing a DQN-based ATSC for two real-world scenarios in Lisbon, Portugal.

Pre-processing

For the purposes of this work, two real-world scenarios in the Lisbon metropolitan area are considered: scenario 1 consists of a single intersection, whereas the second scenario consists of a set of three consecutive intersections. The two scenarios were extracted from OPENSTREETMAP by following the previously described steps. The JOSM editor was used to further crop, rotate and resize the area of interest. The resulting SUMO environment is shown in Figure 3. The considered demands are time-constant and proportional to the number of lanes. The routes are weighted using the previously described synthetic procedure.

MDP formulation and RL approach

We adopt a simple approach where an RL agent controls a single intersection, ignoring the existence of other agents in neighboring intersections (Abdulhai, Pringle, and Karakoulas 2003). In other words, each individual agent is modeled using an MDP that considers only the traffic information in the intersection controlled by that agent.

In each intersection, the signal cycle length is fixed to 60 seconds, and the yellow time to 6 seconds. At the beginning of each cycle, the controller is able to pick the signal plan to be executed throughout the next cycle, from a set of predefined signal plans. In our case, all intersections are composed of two phases. More precisely, the *action space* consists of a discrete set of 7 signal plans $\{$0: (30%, 70%), 1: (37%, 63%), 2: (43%, 57%), 3: (50%, 50%), 4: (57%, 43%), 5: (63%, 37%), 6: (70%, 30%) $\}$, where the first and second elements of each tuple correspond, respectively, to the phase split of phase 1 and phase 2. With this action space definition, adjacent traffic controllers can be easily synchronized and a minimum green time is guaranteed for all phases, easily ensuring that all safety standards are met.

The state s, at cycle c is represented by the tuple (w_1, w_2), where w_p is the cumulative number of vehicles waiting, or navigating at low speeds, in phase p, and can be computed according to:

$$w_p = \frac{1}{K} \sum_{k=0}^{K-1} \sum_{l \in L_p} \sum_{v \in V_l^k} stopped(v,k), \tag{5}$$

where K is the cycle length in seconds (fixed as $K = 60$), L_p is the set of all inbound lanes to phase p, and V_l^k is the set of vehicles on l lane at time k. A vehicle is said to be stopped when it has a very low speed:

$$stopped(v,k) = \begin{cases} 1 & speed(v,k) < \text{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where the used threshold corresponds to 10% of the maximum velocity and all vehicles' speeds are normalized as to belong to the interval $[0, 1]$.

Finally, we use an action-independent reward function defined, for a state $s = (w_1, w_2)$ in intersection i and cycle c with phases P, as:

$$r = -\sum_{p \in P} w_p \tag{7}$$

The reward in Eq. (7) consists of the (negative) sum of the total amount of seconds the vehicles have been stopped during the cycle. We use $\gamma = 0.95$.

Training

We ran 30 independent training runs. Figures 4 and 5 display, respectively, the mean actions selected during training and the observed instantaneous rewards. As it can be seen,

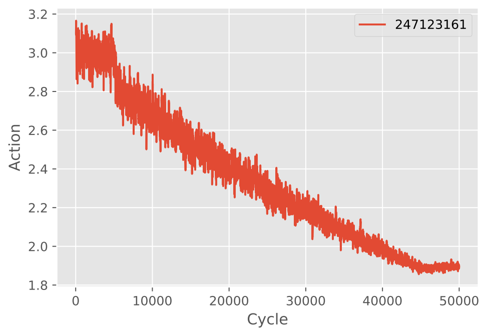

Figure 4: (Scenario 1) Mean actions.

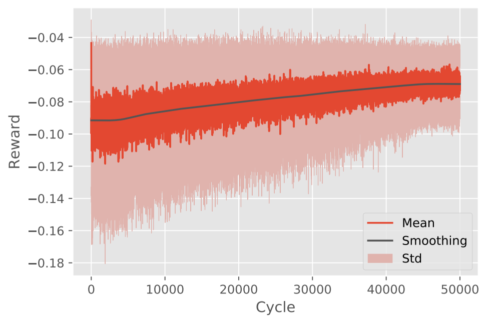

Figure 5: (Scenario 1) Instantaneous rewards.

the agent's actions converge towards lower-indexed signal plans, which can be justified by the intersection's structure (Fig. 3): the horizontal direction serves more vehicles, therefore the agent converges towards lower-indexed actions which allocate a longer period to this phase. As the actions converge it is noticeable an increase in the observed instantaneous rewards.

The outcome of the training stage consists of 30 policies that need now to be properly assessed.

Evaluation

Finally, the performance of the resulting agents is evaluated using a set of performance metrics: travel time, waiting time and vehicles' speed. Tables 1 and 2 display the evaluation metrics for different traffic signal controllers. The actuated controller dynamically extends the current phase, up to a maximum value, if a continuous stream of incoming vehicles is detected.

With respect to scenario 1 (Tab. 1), the results show that the RL controller is able to achieve the highest average speed, outperforming all the other controllers. With respect to the travel time metric, the RL agent is able to outperform the Webster method and equal the performance of both the best static controller and the actuated method. However, the agent is unable to outperform the Max-Pressure controller, but it is important to notice that this method exhibits a higher degree of flexibility in comparison to the RL agent since it´s cycle length is dynamic.

Regarding the second scenario (Tab. 2), the RL agents are again able to achieve the highest average speed. Regarding the travel time, it can be seen that the RL agents are

Method	Speed	Waiting time	Travel time
Static	(6.7, 3.5)	(8.1, 9.9)	(25.0, 12.5)
Webster	(6.6, 3.4)	(8.2, 9.7)	(25.4, 12.4)
Max-pressure	(6.5, 2.8)	**(5.7, 6.5)**	**(23.4, 9.0)**
Actuated	(6.7, 3.4)	(7.8, 10.2)	(24.9, 12.8)
RL controller	**(6.8, 3.5)**	(8.0, 10.1)	(25.0, 12.6)

Table 1: (Scenario 1) Evaluation metrics aggregated per vehicle's trip (averaged over 30 simulations). The first tuple position encodes the mean value; the second tuple position encodes the standard deviation.

Method	Speed	Waiting time	Travel time
Webster	(6.3, 2.7)	(12.0, 12.0)	**(38.6, 14.7)**
Max-pressure	(5.8, 2.2)	**(9.0, 7.0)**	(42.3, 18.6)
Actuated	(5.6, 2.5)	(11.6, 10.5)	(45.5, 22.0)
RL controller	**(6.3, 2.8)**	(12.2, 10.9)	(39.1, 15.5)

Table 2: (Scenario 2) Evaluation metrics aggregated per vehicle's trip (averaged over 30 simulations). The first tuple position encodes the mean value; the second tuple position encodes the standard deviation.

able to outperform the actuated method as well as the Max-Pressure controller, despite the fact that these methods are able to achieve a significantly lower waiting time. This happens due to miscoordination between the adjacent intersections for both the actuated and Max-Pressure methods (due to their acyclic behaviour).

Fig. 6 displays the distribution of the travel time means for scenario 1. As it can be seen, there is a significant overlap in performance between the different methods. Furthermore, despite the fact that the ANOVA test yields a p-value of ≈ 0 (meaning that, indeed, the mean performance of all methods is not the same), the Tukey HSD pairwise tests show no significant difference in mean performance between some methods. Namely, between the actuated, static and RL controllers, the confidence interval on the means' difference either includes zero, or its bounds are close to it.

Finally, Fig. 7 displays the distribution of the vehicles' speed per trip. Interesting to notice the multimodal shape of the underlying distributions, as well as the slight differences between the different methods.

Conclusions

In this paper, we proposed a methodology for the development of RL-based adaptive traffic signal controllers. The proposed methodology comprises 4 steps, all of which necessary to develop, deploy and evaluate an ATSC — simulation setup, problem formulation, training and evaluation. We illustrated the proposed methodology by developing a deep RL-based ATSC that achieves performance on par with established methods from the transportation engineering field. Despite the fact that the flexibility of the agent is constrained in order to met safety standards, the presented results glimpse at the potential of RL-based controllers to contribute to improve traffic congestion and highlight the need for coordination between adjacent intersections. Finally, we

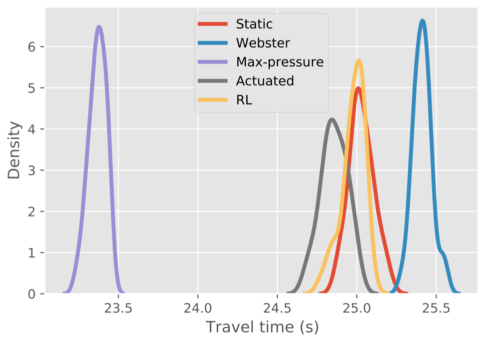

Figure 6: (Scenario 1) Kernel density estimation of the travel time means, computed using 30 samples for each of the methods.

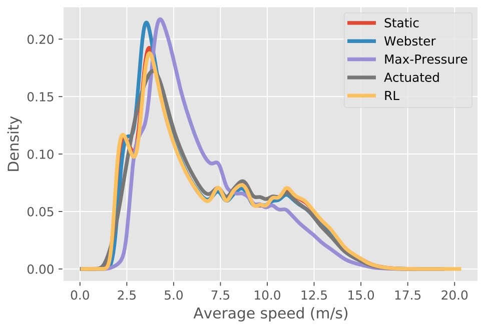

Figure 7: (Scenario 1) Kernel density estimation of the vehicles' speeds, computed using 30 samples for each of the methods.

note that the advantages of RL-methods are more apparent in scenarios comprising bigger traffic networks and more variable traffic patterns, something that could be considered in future work.

Acknowledgments

This work was partially supported by national funds through the Portuguese Fundação para a Ciência e a Tecnologia under project UIDB/50021/2020 (INESC-ID multi-annual funding) and the project ILU, with reference DSAIPA/DS/0111/2018. This research was also partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

Abdulhai, B.; Pringle, R.; and Karakoulas, G. 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129(3): 278–285.

Aslani, M.; Mesgari, M. S.; and Wiering, M. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C* 85: 732 – 752.

Ault, J.; Hanna, J. P.; and Sharon, G. 2020. Learning an Interpretable Traffic Signal Control Policy. In *Proceedings of* *the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, 88–96. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

Becker, U.; Becker, T.; and Gerlach, J. 2016. The true costs of automobility: External costs of cars overview on existing estimates in EU-27. Technical report, Institute of Transport Planning and Road Traffic, Technische Universität Dresden.

Daganzo, C. 2007. Urban gridlock: Macroscopic modeling and mitigation approaches. *Transportation Research Part B* 41(1): 49–62.

Dimitrakopoulos, G.; and Demestichas, P. 2010. Intelligent transportation systems: Systems based on cognitive networking principles and management functionality. *IEEE Vehicular Technology Magazine* 5(1): 77–84.

El-Tantawy, S.; Abdulhai, B.; and Abdelgawad, H. 2014. Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems* 18(3): 227–245.

Genders, W.; and Razavi, S. 2019. An open-source framework for adaptive traffic signal control. *CoRR* abs/1909.00395.

Genders, W.; and Razavi, S. N. 2016. Using a Deep Reinforcement Learning Agent for Traffic Signal Control. *CoRR* abs/1611.01142. URL http://arxiv.org/abs/1611.01142.

Harford, J. 2006. Congestion, pollution, and benefit-to-cost ratios of US public transit systems. *Transportation Research Part D* 11(1): 45–58.

Hilbrecht, M.; Smale, B.; and Mock, S. 2014. Highway to health? Commute time and well-being among Canadian adults. *World Leisure Journal* 56(2): 151–163.

Islam, R.; Henderson, P.; Gomrokchi, M.; and Precup, D. 2017. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *CoRR* abs/1708.04133.

Krajzewicz, D.; Erdmann, J.; Behrisch, M.; and Bieker, L. 2012. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal on Advances in Systems and Measurements* 5(3-4): 128–138.

Liu, Y.; Liu, L.; and Chen, W. 2017. Intelligent traffic light control using distributed multi-agent Q-learning. In *Proc. IEEE 20th Int. Conf. Intelligent Transportation Systems*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602. URL http://arxiv.org/abs/1312.5602.

OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

Prabuchandran, K.; Hemanth, A.; and Bhatnagar, S. 2014. Multi-agent reinforcement learning for traffic signal control. In *Proc. 17th Int. Conf. Intelligent Transportation Systems*, 2529–2534.

Puterman, M. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rodrigues, F.; and Azevedo, C. L. 2019. Towards Robust Deep Reinforcement Learning for Traffic Signal Control: Demand Surges, Incidents and Sensor Failures. *ArXiv* abs/1904.08353. ArXiv: 1904.08353.

Ross, S. 2004. *Introduction to Probability and Statistics for Engineers and Scientists*. Introduction to Probability and Statistics for Engineers and Scientists. Elsevier Science. ISBN 9780080470313.

Schrank, D.; Eisele, B.; and Lomax, T. 2019. 2019 Urban Mobility Report. Technical report, Texas A&M Transportation Institute.

Sutton, R. S.; and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. MIT Press.

Thorpe, T. L. 1997. Vehicle traffic light control using SARSA. Technical report, Department of Computer Science, Colorado State University.

Varaiya, P. 2013. The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections. *Advances in Dynamic Network Modeling in Complex Transportation Systems* 27–66. doi:10.1007/978-1-4614-6243-9_2.

Wang, Y.; Zhang, D.; Liu, Y.; Dai, B.; and Lee, L. H. 2019. Enhancing transportation systems via deep learning: A survey. *Transportation Research Part C* 99: 144–163.

Watkins, C. 1989. *Learning From Delayed Rewards*. Ph.D. thesis, King's College.

Webster, F. 1958. *Traffic Signal Settings*. Road research technical paper. H.M. Stationery Office. URL https://books.google.pt/books?id=c9QOQ4jXK5cC.

Wei, H.; Yao, H.; Zheng, G.; and Li, Z. 2018. IntelliLight: A reinforcement learning approach for intelligent traffic light control. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2496–2505. Association for Computing Machinery. ISBN 9781450355520. doi:10.1145/3219819.3220096.

Wei, H.; Zheng, G.; Gayah, V.; and Li, Z. 2019. A survey on traffic signal control methods. *CoRR* abs/1904.08117.

Whiteson, S.; Tanner, B.; Taylor, M. E.; and Stone, P. 2011. Protecting against evaluation overfitting in empirical reinforcement learning. In *Proc. 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 120–127.

Wiering, M. 2000. Multi-agent reinforcement leraning for traffic light control. In *Proc. 7th Int. Conf. Machine Learning*, 1151–1158.

Zheng, G.; Zang, X.; Xu, N.; Wei, H.; Yu, Z.; Gayah, V. V.; Xu, K.; and Li, Z. 2019. Diagnosing Reinforcement Learning for Traffic Signal Control. *CoRR* abs/1905.04716. URL http://arxiv.org/abs/1905.04716.