Critical Review of Reinforcement Learning Based Adaptive Traffic Signal Control

Miguel Ângelo Mendes Coelho
miguelmendescoelho@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2021

Abstract

Adaptive traffic signal control is a performance-critical task in intelligent transportation systems: Through the use of advanced sensors, controllers can respond to actual traffic conditions at intersections and produce policies capable of mitigating bottlenecks and preventing gridlocks. Reinforcement learning-based multi-agent systems (MARL) provide a natural framework to address the task. In MARL, each agent controls one intersection and can coordinate with its neighbors to attain an optimal network-wide policy. However, it remains a daunting task to compare approaches as there are many possible compositions. This thesis makes a critical comparison across different traffic controllers found in the literature. We show that using non-linear approximators, coordination mechanisms and increasing the observability at each intersection are key performance drivers.

Keywords: Intelligent transportation systems, Reinforcement learning, Multi-agent reinforcement learning, Machine learning

1. Introduction

The most recent growth in population led to an increase in social and economic activities that then resulted in a higher demand for transportation which has outclassed the existing traffic infrastructures who are now incapable of handling everyday traffic congestions.

One way of increasing the traffic capacity is to expand the current road infrastructure. This is normally avoided due to its monetary cost, extensive time commitment and environmental constraints.

Another approach is to create Intelligent Transportation Systems, whose goal is to reduce congestion and provide safety for its users [1].

Classic traffic signal control systems have three main limitations: They use manual labor for data collection, historical data to design signal plans, and make some assume the traffic patterns will hold until the next re-calibration. In contrast, adaptive traffic signal controllers can react to instantaneous traffic conditions using sensors, without manual input or any traffic assumption.

An alternative is to combine Reinforcement learning algorithms into available traffic data sensors. The advantage is threefold: The system incorporates historic traffic patterns and reacts, taking the best long-term action into account. The system can adapt automatically to new traffic patterns as they emerge. And can optimize for plans for more than one intersection.

Reinforcement learning (RL) is usually modeled by *Markov decision processes* (MDP) [2]. MDPs describe sequential decision-making problems where an agent interacts with an environment by observing a state and making a decision. In response, the environment transitions to another state and emits a reward signal based on the relative improvement of the system. The agent's objective is to search for a policy that maximizes the average cumulative reward over time.

The flexibility of the MDP framework comes with a drawback; there are many different ways to express it. Specifically, there are many state, action and reward definitions. Wei, *et al.* [3] provides a comprehensive survey on all the state formulations and action schemes used in the literature.

Multi-agent reinforcement learning (MARL) provides an alternative to a centralized system controlled by a single agent, when achieving centralized control might be either costly or non-viable. In the particular case of RL-based adaptive traffic signal control (RLATSC), centralized control entails controlling many intersections, leading to a high computation cost due to the scale of the state and action spaces.

As MARL provides a way to decompose a task, the system can handle larger problems by adding more agents and scale quickly, while learning

agents benefit from decreased learning cycles due to communication and information sharing.

The straightforward way to approach MARL is building many independent learners (ILs) who learn oblivious to their neighbors. But, the dynamics of the environment may become unstable, *i.e.*, causing a *non-stationary effect*. Agents cannot differentiate environment changes that are caused by their actions, by other agents' actions, or even intrinsic changes from the environment itself. This can prevent learning from happening altogether.

As with the single-agent case, there are many ways to define multi-agent systems, including not only the MDP formulation but also other factors, such as, the type of discretization used, if coordination methods are applied and how much information is given to each system during training and execution. Unless the many works in the literature are compared along the same axes, researchers are left unable to assess the aspects that make multi-agent reinforcement learning systems work.

Thus rises the main question of this work: How do different methods of discretization, coordination and observability affect traffic controllers when run on a homogeneous simulation environment?

2. Contributions

The major contributions of the current work are to:

- Prepare the traffic simulator and create a set of scenarios that will be used to train and evaluate the controllers.
- Define the MDP formulation that allows for a meaningful comparison between the different traffic control methods.
- Develop baseline traffic controllers, in particular: a random controller, a fixed-time controller, a Webster controller [4] and a Max-pressure controller [5].
- Develop reinforcement learning-based traffic controllers, in particular: an Actor-Critic model [6], a DQN model [7] and three different implementations of the MARLIN algorithm [8], based on Fictitious Play.
- Analyse how do different methods of discretization, coordination and observability affect the baseline and reinforcement learning-based traffic controllers, using a set of evaluations metrics to access the performance of each controller.

3. Related Work

Traffic signal controllers have evolved over the years to better respond to the growing traffic flow by making use of the technological advancements in computation and the increasing data collected from sensors.

Starting with fixed timing controllers, that normally define different cycle profiles and alternate between them depending on the observed traffic flows.

Webster [4] is a closed form method that calculates the phase split ratio, using a fixed cycle length, that minimizes the travel time, assuming the traffic flow is uniform during a certain time period.

GreenWave [9] is a method that implements coordination by optimizing the timing offsets of nearby intersections to minimize the number of stops for vehicles moving in a certain direction.

Maxband [10] also optimizes timing offsets by minimizing stops but now considers a two-way environment, using linear programming.

Later traffic controllers started using models that used sensor data to optimize different traffic metrics to better adapt to the changes in traffic flow.

Max-pressure [5] aims to balance the queue length of neighboring intersections. The pressure of a phase is defined as the difference between the queue length of incoming and outgoing lanes. It is shown that Max-pressure maximizes the throughput of the system if the pressure of the phases is minimized.

Tubaishat, *et al.* [11] use a wireless sensor network to create a decentralized system that improves the localized flow and coordination between neighbor traffic lights.

These systems generally outperform fixed-timing controllers but were tested in very simple scenarios with a single intersection or with very restricted traffic assumptions, like uniform traffic flow and cannot adapt well to real world city-level traffic.

3.1. Reinforcement learning

Different learning models have been explored to build traffic signal controllers. However, it is hard to compare the proposed solutions and results since the problem definition varies greatly between the literature.

3.1.1 Modeling

The most used representations for the states $\mathcal{S}$, actions $\mathcal{A}$, and rewards $\mathcal{R}$ in the literature are described in this section.

In the context of the traffic control problem, the environment is the network of roadways that approach the intersections and the vehicles that traverse these roadways. Since the traffic problem has a complex and stochastic environment, most works rely on simulators that accommodate the RL framework. Even though in these simulators the environment is fully observable, using all of the

data available might prove difficult when trying to apply the model to the real world.

Most works use state representations that are derived from data that real sensors capture in a typical urban roadway. The definition of the environment state in the literature is a combination of one or more of the following parameters: Cumulative delay, phase information, queue length, volume, vehicle position, vehicle speed and saturation.

These state definitions can be simpler or more complex depending on the number and complexity of the metrics used. A more complex state representation is built in hopes to better represent reality, but may lead to big state spaces that can hinder the training performance, without actually boosting the effectiveness of the model. It should be noted that some of these values are harder to obtain than others. For example, vehicle speed and position might be difficult to accurately obtain from sensors positioned in the roadway.

In the traffic control problem, the set of actions, $\mathcal{A}$, is the set of phase controls of the traffic controller.

Examples of actions used in the literature are: Set Phase duration, Choose Phase ratio, Keep or change phase and Choose next phase.

Regarding the reward function $\mathcal{R}$, in the traffic control problem the objective is, ultimately, to minimize the travel time of all vehicles in the network. However, this metric can be hard to obtain for every vehicle since most times the vehicle destination is unknown.

Other metrics that can be more easily measured after each action is taken, in the hope that minimizing them also minimizes the travel time of all vehicles, such as, cumulative delay, queue length, volume and vehicle speed.

The reward function can be defined as a weighted sum of one or more of the previous metrics, this can be tricky as there is no straightforward way of tuning these weights. Thus, the weights are generally tuned empirically or a single metric is used.

3.1.2 Classic reinforcement learning

Wiering [12] used a model-based approach that operates in 6 intersections, where each controller receives the discretized position and destination of each car. Even though the RL-controllers outperform more simple controllers, this model assumes that cars can communicate with traffic light controllers, which is currently unfeasible. The network is also simplified since every street has the same number of lanes, resulting in an unrealistic homogeneous traffic pattern.

The previous work was extended in [13] by adding the direction of each car to the state and tested on a bigger network with 36 roads and 15 intersections. The RL controllers outperformed fixed controllers and co-learning between cars and traffic lights minimized waiting time by 50%.

Some works [12, 14, 15, 16] have tried model-based approaches but most of the research community adopts model-free methods since it is difficult to fully model $\mathcal{P}$, given the natural unpredictable behavior of human drivers.

Most works using model-free methods rely on algorithms like Q-learning and SARSA to learn a optimal traffic control policy.

Thorpe at al. [17] created a model-free system using SARSA and compared the performance of 3 different state representations. The RL model outperformed fixed time and greatest volume controllers, independently of the state representation used.

Zhou, *et al.* [18] compared a SARSA model with a fixed time approach. The authors used the saturation ratio between current and maximum flow and vehicle speed, both normalized in 7 discrete values for a total of 49 states. The SARSA model had almost half travel time when compared to an offline model when the traffic flow was low but had negligible difference in high traffic flow.

El-tantawy, *et al.* [19] compared different learning methods, state, action and reward representations and action selection methods. Where, TD(λ) with eligibility traces outperformed Q-learning and SARSA and RL-controllers generally outperformed pre-timed controllers regardless of the state representation. For action selection during learning, the best approach was a mix of ε-greedy and softmax.

Earlier reinforcement learning based controllers are applied to a single intersection since the state space grows exponentially with the number of intersections that are controlled.

3.1.3 Multi-agent reinforcement learning

In a multi-agent setting, each agent controls one intersection in a traffic network with several intersections. This way, the explosion of the state space is minimized by making each agent operate on a small partition of the environment.

Camponogara, *et al.* [20] created a multi-agent system based on Q-learning as a distributed stochastic game in a 2 intersection network and compared it with a random and LQF[1] policy. The proposed model had significant performance gains over the other two policies. However, the agents did not collaborate and the proposed scenario was very simplistic.

[1] Longest Queue First

Aslani, *et al.* [6] uses a well-known reinforcement learning model, actor-critic, and a classical discrete function approximation method, *tile coding* [2] to control 50 junctions in downtown Tehran.

Mnih, et al. [7] introduce the *deep Q-network* (DQN) in the domain of Atari games. The approach is characterized by the use of a running replay buffer to store experiences that serve as input to the neural network. DQN is soon adapted for ATSC, controlling one junction [21] and outperforming common baselines.

Since the actions of one agent can affect other agents, having self-interested agents that only try to maximize the gain in their intersection may lead to better local performance but worse global performance. Thus, some form of collaboration or information sharing between agents is used.

The naive approach would be to simply add information about all other intersections to the state space. This leads to an exponential growth with the number of intersections and is unfeasible in larger networks. Thus, the main challenge in the multi-agent setting is to implement coordination and information sharing between the agents while maintaining a state-space with manageable size.

Kuyer, *et al.* [22] use the Green Light District simulator to design a vehicle-based model. The system achieved coordination by using the Max-Plus algorithm. The model was compared to the Wiering [12] model and Steingröver's [23] extension. The devised model outperformed the others in both small and big networks.

Van der Pol [24] applied a DQN algorithm with vehicle position as input. For a single intersection, the agent had better stability and performance when compared to a baseline model. The author then applied batch normalization, prioritized experience replay and double Q-learning which created an improvement in stability and performance. Transfer learning and the Max-Plus algorithm were also applied. This model had better performance than the vehicle-based approach introduced by Wiering [12] and the Kuyer's extension [22] that used the base Max-Plus algorithm in Wiering's model.

Another joint action learning approach is using a variant of *fictitious play* where an agent selects its action according to the estimates their opponents' reaction to its state. Aiming to reach a steady-state condition, or an equilibrium where no agent has an incentive to deviate.

El-tantawy *et al.* [8] introduces a variant of fictitious play where agents store a model from their neighboring agents' actions. By counting the relative frequency that a particular neighbour has picked a certain action, the agents can update their action model. The method uses tabular Q-learning and was shown to control traffic in downtown Toronto, composed of 59 intersections.

Collaborative multi-agent systems are a way to handle the curse of dimensionality when considering complex networks and are shown to outperform fixed-timing, single RL agents and non-collaborative multi-agent RL models. However, most works rely on either directly adding information about other agents to the state or take advantage of coordination graph approaches, like the Max-plus algorithm, that exploits the space locality of the agents. The actual effectiveness of these coordination methods is hard to grasp since each work defines a different set of state-action representations and testing scenarios.

This work is inspired on [25, 26] where comparisons are made between traffic controllers with a fixed MDP formulation. To ensure fairness, a methodology inspired by Varela [27] is used for building experiments and making evaluations.

4. Methodology

Following a rigorous methodology is essential in making experiments reproducible and the results comparable. The methodology used is slightly adapted from Varela [27], a methodology for Reinforcement Learning based Adaptive Traffic Signal Control for multiple coordinating agents. Whereas the original methodology for independent learners consisted of four steps, in this work, step number two is expanded into two distinct parts: MDP formulation and RL method. The five steps are as follows: the simulation setup, the MDP formulation, the RL method, training and evaluation (Fig. 1).

Since the MDP defines the optimization problem, any meaningful comparison between RL-based methods, must be have the same base MDP. Moreover, the MDP formulation has a decisive impact on the performance of said methods, as it can be demonstrated by holding the learning algorithms fixed and changing the base MDP formulation [28, 29]. In this work, a base MDP is fixed and several baselines and RL-based methods are compared, to test distinct function approximators, coordination methods and observation scopes.

4.1. Simulation setup

The simulation setup consists on the choice of traffic simulator used, where the agents will train in a realistic simulation and attempt to learn effective traffic control policies, the topology of the networks used in said simulator and finally, defining traffic demands.

4.1.1 Traffic simulator

In contrast to traffic macro-simulators, that simulate traffic flow as a whole, traffic micro-simulators like

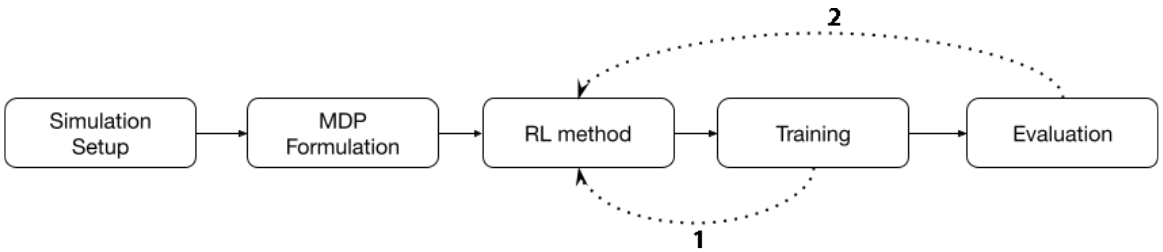


Figure 1: Diagram illustrating the proposed methodology, composed of five stages. Solid arrows denote the main development flow, whereas dashed arrows denote the iterative process of model tuning.

Paramics [30], SUMO [31], AIMSUN[2] and CityFlow [3][32] generate vehicle dynamics, *i.e.*, each vehicle property, such as position, speed, acceleration, route and emission rates are simulated for each vehicle. These have been used to evaluate current traffic controllers and prototype new ones. In the reinforcement learning context, they can be used to model the environment where the agent learns the traffic policy.

This work uses the CityFlow[3] microsimulator [32] since it is open-source, simple to use and more efficient than other widely used simulators like SUMO [31].

Every simulated time step corresponds to one second. However, the learning agents only observe and take decisions every 10 seconds. This is due to the fact that agents can take an optimal action at a certain state but still receive a bad reward since the vehicles haven't yet moved out of the intersection, due to the observation rate being too frequent or the yellow light being in effect.

For all experiments there is a minimum of 5 seconds and a maximum of 85 seconds of green time and a fixed yellow time of 5 seconds. This means that regardless of the agent's action, the phase will always remain the same until it reaches a duration of 10 seconds and it will always switch after it reaches a duration of 90 seconds.

4.1.2 Road networks topology

Networks can be either synthetic or extracted from real world locations. Available open source services, such as OPENSTREETMAP[4] , 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 this work, all networks use geospatial data from OPENSTREEMAPS to build the configuration files to be used by the simulator. The following steps are required: **(i)** Extract the region of interest from OPENSTREETMAP and open the resulting file with the JOSM[5] editor, an extensible editor for OPENSTREETMAP files, in order to fine-tune the network; **(ii)** Convert the edited OPENSTREETMAP file into a SUMO network format using the `netconvert`[6] tool; **(iii)** Open the resulting SUMO network file with the `netedit`[7] tool, a graphical network editor for SUMO, in order to ensure that all intersections are properly setup; **(iv)** Convert the sumo network file to a CityFlow compatible network file using the converter tool provided by CityFlow.

For this works' experiments, two scenarios depicting an arterial road, 1x3 road mesh and a 3x2 network grid extracted from the city of Lisbon, represented in Figure 2

Figure 2: Grid network with 6 traffic controllers.

All traffic lights in both networks are modeled with two phases: One allows right-turn and through for vertical movement and the other allows right-turn and through for horizontal movement.

4.1.3 Traffic demands

In the domain of ATSC, the traffic demands that are simulated can be either derived from real-world data or synthetically made.

[2] `http://www.AIMSUN.com`
[3] `https://cityflow-project.github.io`
[4] `https://www.openstreetmap.org`
[5] `https://josm.openstreetmap.de/`
[6] `https://sumo.dlr.de/docs/netconvert.html`
[7] `https://sumo.dlr.de/docs/netedit.html`

Real-world data allows for the creation of realistic traffic demands that match real observations, shortening the gap between the simulator and the real-world, where the implemented traffic controllers are to be deployed at some point. However, this complicates the setup process due to the fact that this data needs to be validated before being used and is usually network specific. This data can also be hard to obtain, noisy or even unavailable. As such, data-driven traffic demands are outside of the scope of this work.

For all experiments in this work, a constant synthetic traffic demand is used, with constant weighted route emissions probabilities, where routes with large avenues receive greater weights.

The routing is done according to the following guidelines: They are the shortest paths connecting incoming and outgoing edges from the road network. And, if a generated path results in at least a loop, where the vehicle needs to change cardinal directions at least three times, the path is excluded from the final selection. This filtering is necessary since those paths are extremely unlikely to happen in real life and may cause simulation gridlocks.

4.2. MDP formulation

An MDP consists of a state feature, a reward signal, an action schema and an observation scope. In MARL, a group of collaborating agents can be defined by a DecMDP with N agents:

$$(\mathcal{S}, \mathcal{A}^{(n)}, \mathcal{Z}^{(n)}, \mathcal{P}, \mathcal{O}^{(n)}, \mathcal{R}, \gamma),$$

- **State space** $\mathcal{S}$: $s \in \mathcal{S}$ is the state at time t, consisting of features of the incoming approaches of an intersection. In this work, it is described by a feature map $\phi(s)$ composed of an internal state and the data on its incoming approaches:

  $$\phi(s) = (x_g, x_t, x_0, \ldots, x_p, \ldots x_{P-1}).$$

  The internal state is defined by the index of the current green phase, $x_g \in \{0, 1, \ldots, P-1\}$, where P is the number of phases, and the time since this phase has been active, $x_t \in \{10, 20, \ldots, 90\}$. The feature x_p on the incoming approaches of any given agent n at phase p is defined by the cumulative delay:

  $$x_p = \sum_{v \in \mathcal{V}_p} e^{-5(v/v_p^*)}, \tag{1}$$

  where v_p are the velocities of the vehicles in the incoming approaches of phase p for the agent and v_p^* is the speed limit for phase p. If every vehicle travels at the speed limit or there aren't any cars then there is no delay. As a vehicle travels slower, the delay increases until it hits full stop ($v = 0$) and its delay becomes its maximum, 1. This choice was heavily influenced by the research done by Pedro [26], where several state features are compared. The cumulative delay has three main benefits: **(i)** It packs the information of the number of cars and their velocities in one single number, having a reduced state space when compared to other features; **(ii)** It doesn't involve any assumptions w.r.t how slow does a vehicle need to be to be considered stopped; **(iii)** Finally, the negative exponential helps to balance situations where there are dominating influx in one phase.

- **Action space** $\mathcal{A}$: At time t, each agent n must take an action $a_t^n \in \mathcal{A}^n = \{0, 1\}$. The action space is homogeneous and binary where 0 means to keep the current phase active for the next decision step and 1 means to switch to the next phase. The action schema has the following constraints 5 seconds of yellow time, 5 seconds of minimum green time, 85 seconds of maximum green time and agents decide every 10 seconds. Hence, the signal plans are cyclical with variable cycle length from a minimum of 20 seconds and a maximum of 180 seconds for the two existing phases.

- **Observation space** $\mathcal{Z}$: Each agent n can observe only a partition of the state $z_t^{(n)}$ – $\bigcup_n^N z_t^{(n)} = s_t$. $z_t^{(n)}$ is totally defined by the agent's internal state and the data on its incoming approaches. The feature space for agent n becomes $\phi((z^{(n)})_{n=1}^N)$.

- **Reward** $\mathcal{R}$: The reward collected by an agent for selecting action a at state s is given by:

  $$\mathcal{R}(a, s) = -\sum_{p=0}^{P} x_p, \tag{2}$$

  where p is the phase indicator, x_p is the cumulative delay feature at phase p of agent n. The reward is actually a penalty where each agent seeks to minimize the delay on the incoming approaches.

- **Transition probability** $\mathcal{P}$: In RLATSC its usual to set $\mathcal{P} = \emptyset$, meaning that the transition probabilities are not explicitly estimated by the reinforcement learning algorithm. We must use *model-free* [2] reinforcement learning methods, in which the agents learn only by interacting with the environment.

- **Observation probability** $\mathcal{O}$: Following the same logic, $\mathcal{O} = \emptyset$.

- **Discount factor** γ: The discount factor $0 < \gamma \leq 1$ shapes the agents' actions towards either minimizing the penalty on the short or long term. When $\gamma \to 0$ the agent acts myopically, conversely if $\gamma = 1$ then the agent is indifferent in relation to instantaneous rewards and future rewards. In this work, $\gamma = 0.98$ is used.

The choices made in this MDP formulation were driven by previous research by P. Santos, *et al.* [29]. For a more detailed survey on the different MDP formulations used in the literature, refer to Wei [3].

4.3. Reinforcement learning methods

The Reinforcement learning methods consist in learning algorithms with different function approximation methods, coordination methods and observability scopes.

In this work, the coordination between agents is accomplished by using the original MARLIN algorithm, defined by El-tantawy *et al.* [8] and several variations of it.

It is worth noting that: **(i)** The original MARLIN algorithm receives a discrete state space, thus, the state defined in the previous MDP formulation needs to be discretized. **(ii)** In this algorithm, each intersection needs to share their states and actions during training and their states during the execution.

Specifically, five learning algorithms were trained and tested:

An Actor-Critic model, following the implementation of Aslani, *et al.* [6] with a 0.9 critic learning rate (α), a 0.3 actor learning rate (β) and a trace decay (λ) of 0.55. Since the state is continuous and this is a tabular method, tile coding is used with 5 tiles. For example, an observation of (1, 40, 5.0, 20.0) would become (1,2,1,4) after discretized.

A DQN model [7] using a neural network with one hidden layer, Double Q-learning, experience replay and batched learning.

Three variations of the MARLIN algorithm [8], where each agent saves a Q-table and a policy estimation for every neighbour, specifically, an implementation where delay is simply rounded, an implementation using tile coding and an implementation that replaces the tabular Q-learning and policy estimator with deep Q-networks.

A few baseline models will also be implemented to compare against the RL-based methods. Specifically, a random controller, a fixed time controller, a Webster controller [4] and a Max-pressure controller [5] will be implemented.

4.4. Training

During training, several simulation and algorithm specific procedures depend on random number generators. Simply changing the random seed of said generators can cause statistically significant differences in the performance of the implemented traffic controllers. Thus, 30 independent training runs are seeded for every experiment and the results are then averaged for each controller. This random seeding also makes every experiment fully replicable.

Since the learning methods have exploration and exploitation phases, during the simulation, a gridlock can happen in the network, preventing the vehicles to move through the road network. This can happen more often when actions are selected randomly. When a gridlock happens, the agents stop learning and the simulation essentially halts. To avoid them, the RL task is made episodic to ensure that unfavorable outcomes do not carry on indefinitely. Each episode runs for a total of 6 hours (21,600 seconds) and every individual run trains for 80 episodes, totaling a simulated training time of 600 days.

The reward increase and loss decrease are also monitored to determine good algorithm parameters. If rewards fail to increase or loss does not drop significantly then it's not worth to proceed to the evaluation phase and the RL method must be re-configured (dashed arrow 1 in Figure 1).

4.5. Evaluation

Similar to the training phase, every evaluation needs to have multiple independent runs to allow for significant results and reproducibility. Every training run performed is tested during one episode's length (6 hours) and also with set, unique random seeds, totaling a simulated evaluation time of 30 episodes (7.5 days).

The key metric used during evaluation is the average travel time. Agents usually cannot observe the travel time instantaneously – as it depends on the route the vehicles are traveling, on complex inter-vehicle dynamics and on future decisions made by other agents along their route, thus, additional metrics, such as, vehicle speed and number of stops are also evaluated for each model.

4.5.1 Hyper-parameter tuning

While the training evaluations determine if the agents are learning, the objective of the evaluation run is to confirm that the average behavior of the policies of a given model is satisfactory. It's often the case that, during this evaluation, many rounds of fine tuning must ensue to arrive at a satisfactory parametrization.

Similar to the training run, if the travel time observed during the evaluation run of each method is unreasonable, the RL method must be reconfigured again (dashed arrow 2 in Figure 1). After the RL methods are stabilized, the results of the best models can now compared and analysed.

4.5.2 Performance analysis and comparison

To compare the different baseline and RL-based controllers, metrics such as, rewards, speeds, number of stops, travel time, number of vehicles and waiting time are measured.

First, these metrics are compared by analysing their mean and standard deviation values. This analysis provides an overview of the performance of each traffic controller.

Secondly, the distribution of these metrics can also be explored to determine the significance of the previous analysis and to determine some controller properties, such as, fairness. For example, one traffic controller might learn a policy where the road with most lanes gets the maximum green time and conversely, the road with least lanes gets the minimum green time. This policy might obtain a reasonable average travel time, but it is very unfair.

The policies can also be analysed by plotting the average actions of each agent over a series of time steps. This describes the frequency that each intersection changes phases.

A concrete analysis of the results of the implemented controllers using the above methods is done in Section 5.

4.6. Software

As mentioned before, the traffic simulation is done by the CityFlow micro-simulator[8] [32]. The tensor computation used in the deep RL methods is done by the PyTorch library and the extra data management that is not handled by PyTorch is done using the numpy and pandas libraries.

The Tensorboard toolkit was also used to provide measurements and visualizations of the learning algorithm's during the training phase.

5. Results & discussion

This chapter will present and discuss some of the results obtained from the experiments in the arterial and grid networks.

5.1. Arterial network

The first network is an arterial network with 3 intersections. The network topology causes an imbalance of traffic flow since the horizontal road has more lanes than the other three and the demands are created depending on the number of lanes of each road.

[8] https://cityflow-project.github.io

Method	Travel Time	Speed
Random	17.693±18.127	7.485±3.858
Static	13.882±12.585	8.644±3.660
Webster	13.665±12.453	8.670±3.647
Max-Pressure	**10.872±8.917**	**8.867±3.004**
ACAT	13.976±15.210	8.277±3.441
DQN	11.716±7.861	8.417±3.100
MRL-Round	12.622±12.230	8.358±3.277
MRL-TileC	**10.788±7.290**	8.505±2.924
MRL-DQN	10.921±6.856	**8.631±2.879**

Table 1: Arterial network results.

Regarding the baselines, every traffic controller outperforms the random controller, as it is expected. The Max-Pressure controller greatly outperforms the static and Webster controllers in every metric. Since both the static and Webster controllers are periodic, they require a fixed cycle length (in this work, 60 seconds) and are thus less reactive when compared to the Max-Pressure or RL-based controllers. Since these aperiodic controllers follow the keep/change action schema, they can cycle through the two phases more frequently and vehicles are able to cross the intersections faster, being able to reach cycle times of 20 seconds.

First, a comparison between the performance of the models that have different discretization methods can be done both in the independent and coordinated learners.

When comparing the independent learners, the DQN model achieves an average travel time ≈ 2.3 seconds lower than the Actor-critic model. One factor that can influence this difference is the fact that the Actor-critic controller uses a discrete discretization method (tile coding) to the state space, while the DQN controller uses the continuous state directly, applying a non-linear approximator (neural network) to estimate the Q-values.

When comparing the different coordinated learners, the MARLIN implementation that uses rounded delay performs much worse than the other two, while the tile coding and continuous implementations have similar performance. Since the rounded delay implementation simply transforms the continuous state space by rounding the delay to the nearest integer, the resulting state space is too big for the agents to learn effectively. This is due to the coordination mechanism applied that requires one Q-table for every unique link, each indexed by the concatenated state of two agents, where each table cell contains the Q-values for the joint action of both agents. For example, in the arterial net-

work, where there are 4 unique links, 10 possible phase durations, the delay ranges between 0 and 21 in the first phase and between 0 and 42 in the second phase and there are 4 possible joint actions between two agents, the algorithm needs to learn 4,978,713,600 different Q-values, a much higher value when compared to the tile-coding implementation with 5 tiles that only needs to learn 1,000,000 different Q-values.

Thus, we can conclude that, regardless of the method of coordination used, the tabular methods have higher average travel time and lower average vehicle speed when compared to methods that use non-linear approximators, such as deep neural networks.

Secondly, a comparison between the performance of the models that have different coordination mechanisms (or lack of) can be done in the RL-based controllers.

Even with the size of the state space employed by the MARLIN-Rounded controller, the model still outperformed its non-coordinated counter-part, the Actor-critic controller, by ≈ 1.36 seconds lower average travel time. The tile coding and continuous MARLIN implementations both outperformed the DQN controller, where the tile coding MARLIN implementation achieves a lower average travel time of ≈ 0.93 seconds when compared to the DQN controller.

This shows that, in this network, the use of explicit coordination mechanisms allow controllers to have a lower average travel time and higher average speed when compared to methods that use no coordination, such as the independent learners and baselines.

It is worth noting that, even though the MARLIN implementation using tile coding outperforms every other controller, the Max-Pressure baseline has a very similar average travel time, only being ≈ 0.09 seconds higher when compared to the MARLIN controller. It also has the highest average speed when compared with any other method. This can be explained by the observability that this controller has when compared to the other controllers.

In all the other baselines controllers and independent learner controllers, the agents in each intersection act by only using information that comes from the incoming lanes of that specific intersection. For every individual intersection, the Webster controller uses the number of vehicles and the independent learners use the delay in each incoming lane. The observability of these controllers is smaller when compared to the Max-Pressure and MARLIN controllers.

The Max-Pressure controller minimizes the phase pressure, that is defined as the difference between the queue length of incoming and outgoing lanes, thus, using information not only from the incoming but also the outgoing approaches. It has also been shown in other works [26] that this baseline can outperform other baselines and even RL-based methods.

The MARLIN algorithm only uses information about the incoming approaches. However, each agent selects actions using the delay of its own intersection and the neighbouring intersections, thus, requiring agents to share information during executing but having more observability than the independent learner controllers.

We can conclude that for this network, when using function approximation methods, non-linear approximators outperform discrete approximators, applying coordination methods to the multi-agent system allows controllers to obtain better performance when compared to non-coordinated ones and finally, controllers with higher observability obtain better performance when compared to controllers that have lower observability.

5.2. Grid network

The second network is a grid-like, 2x3 mesh, that also includes roads with a different number of lanes. This network is represented in Figure 2.

Method	Travel Time	Speed
Random	26.388±25.875	7.103±3.798
Static	20.372±15.501	7.962±3.725
Webster	20.496±15.742	7.953±3.719
Max-Pressure	**16.043±11.212**	**8.452±3.109**
ACAT	21.093±22.830	7.783±3.451
DQN	**16.772±10.026**	**8.112±3.003**
MRL-Round*	———±———	———±———
MRL-TileC	18.191±13.946	7.915±3.241
MRL-DQN	17.184±9.623	7.858±2.955

Table 2: Grid network results.
*Too computationally extensive to finish.

Similar to the previous network, every traffic controller outperforms the random controller and the Max-Pressure controller outperforms the static and Webster controller. Further reinforcing that controllers that are more reactive, obtain better performance when compared to less reactive controllers, such as aperiodic controllers.

Regarding the different discretization methods, the DQN model also outperforms the Actor-critic model in both average travel time and average speed, showing that a non-linear approximator is more efficient than a discrete discretization method when tested in a larger network.

For the coordinated learners, the MARLIN im-

plementation that used rounded delay proved to be too computationally extensive to finish the training phase. Following the same calculation done for the previous network, in the grid network, the MARLIN-Rounded controller needed to learn 59,660,697,600 different Q-values. Even though the MARLIN-Continuous implementation has a lower performance than the MARLIN-TileCoding in the simpler arterial network, when provided with a bigger and more complex environment, such as the grid network, the MARLIN-Continuous achieves an average travel time $\approx$ 1 second lower than the MARLIN-TileCoding implementation, showing that the use of a non-linear approximator, such as a deep neural network, can outperform a discrete discretization method, such as tile coding, when the complexity of the environment scales.

As in the arterial network, regardless of the method of coordination used, non-linear approximators have higher performance when compared to tabular methods that use discrete function approximators.

Regarding the different coordination mechanisms in the RL-based controllers, both MARLIN implementations outperform the Actor-critic controller. However, the DQN controller outperforms both MARLIN implementations, achieving an average travel time $\approx$ 0.41 seconds lower than the MARLIN-Continuous implementation. This discrepancy might be explained due to the lack of training steps on a more complex network. For example, by analysing Figure 3, displaying the average actions taken and average vehicle speeds on the MARLIN-TileCoding controller during training, we can observe that the policy is still very unstable and the average speeds are steadily increasing when the model finishes the training phase.

In this grid network, the Max-Pressure baseline manages to outperform every other baseline and RL-based controller in every metric except throughput. Achieving an average travel time of $\approx$ 0.73 seconds lower than the best independent learner and an average travel time of $\approx$ 1.14 seconds lower than the best coordinated learner. Showing that higher observability, achieved in this particular controller by using environment information extracted from the outgoing lanes, leads to better performance.

Concluding, in this network, non-linear approximators outperform discrete approximators, regardless of the coordination method used. The relevance of the coordination mechanisms was inconclusive due to the fact that the learning models did not train enough. And controllers with higher observability obtain better performance when compared to controllers with lower observability.

6. Conclusions

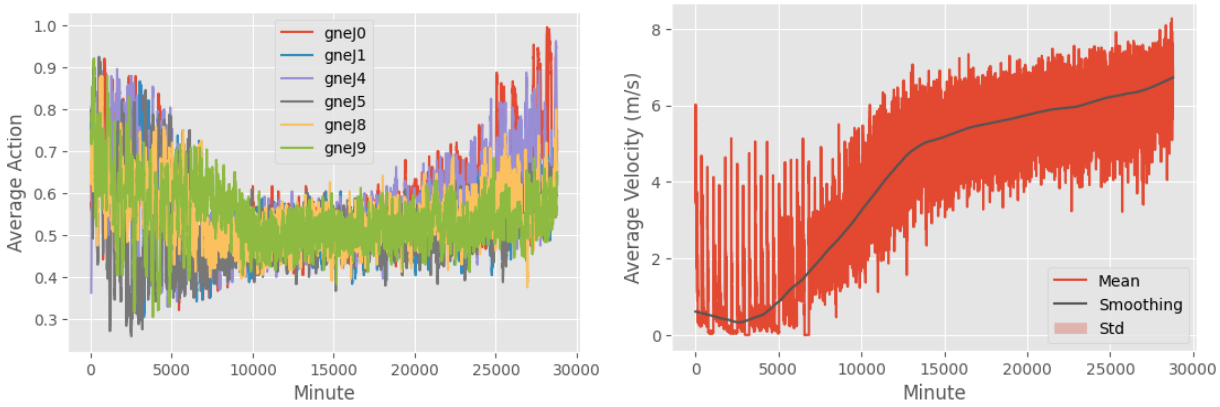
This thesis provides a comparative study of varied traffic controllers with different discretization methods, coordination mechanisms and observability scopes, under a homogeneous environment and a fixed MDP formulation. Every controller follows a multi-agent approach, where each agent controls one intersection by observing a partial environment state and choosing to maintain the active green phase or to switch to the next phase. This work compares a group of baselines and RL-based controllers, specifically, two independent learners using an Actor-critic model and a DQN model, three different implementations of the MARLIN algorithm and five baselines to compare to, including, a random controller, a static controller, a Webster controller and a Max-pressure controller.

This comparative study follows a rigorous methodology, where first, the simulator is set up and the networks and traffic demands are defined, then, a fixed MDP formulation is created. Afterwards, the hyper-parameters of the RL-based methods are tuned using training and evaluation phases. Finally, after being tuned, they are compared between each other and between the defined baseline controllers.

The results obtained by the current literature in the domain of RL in TLC use different methodologies, including different traffic micro-simulators, MDP formulations and result analysis. Thus, they are very hard to compare to newly created controllers or even between the existing literature. It is also hard or even impossible to fully reproduce the traffic controllers and results in some works given the vague methodology that is presented in them. Hence, following a rigorous methodology, such as the one in this thesis, contributes to the application of RL in TLC by allowing different works to be compared, fully reproduced at a later date and even possibly extended.

The simulator was set up and the process of obtaining the real-world networks used in this work is defined, step by step, so it can be fully replicated and potentially extended to any other arbitrary real-world network. Then, a fixed MDP formulation is defined following the results found by Pedro [26], where it is shown that the cumulative delay representing the state and reward is found to have good performance, when compared to other definitions, while packing the lane information into a single number, minimizing the state space. It is also shown that aperiodic controllers usually outperform periodic controllers since the former can react at a faster rate to traffic flow changes, thus, in this work a keep/change action schema is used in all traffic controllers that allow it.

After the simulation is setup, the MDP is de-



(a) Average actions taken by the MARLIN-TileCoding model during training in the grid network.

(b) Average vehicle speed in the MARLIN-TileCoding model during training in the grid network.

Figure 3: Average Action and vehicle speeds in the MARLIN-TileCoding implementation during training in the grid network.

fined and the hyper-parameters of the algorithms are tuned, the results of the evaluations were compiled and compared in two distinct networks, a 1x3 arterial network and a 2x3 grid-like network, focusing on three different axes: The function approximation method used. The coordination mechanism used. And finally, the observability that each controller has.

Results show that, when comparing the function approximation method used, in both networks, controllers that use non-linear function approximators, such as deep neural networks, achieve higher performance than controllers that use discrete discretization methods, such as tile coding, regardless of the coordination method used. These controllers achieve up to $\approx 25\%$ lower average travel time, when compared to its discrete counterparts. It is also shown that for the smaller network, the tile coding and continuous implementations of the MARLIN algorithm have similar performance, but for the bigger, more complex network, the continuous implementation outperforms the discrete tile coding implementation.

When comparing uncoordinated and coordinated controllers, for the first network, the coordinated MARLIN controller that uses tile coding is the best performing controller, followed by the Max-Pressure baseline and the continuous MARLIN controller, showing that a controller that uses an explicit coordination mechanism can outperform every baseline and every independent learner that was implemented, being able to achieve an average travel time of $\approx 8\%$ lower when compared to the best independent learner. For the second network, the coordinated methods only outperformed one of the two independent learners due to lack of training steps.

Finally, the observability of the methods tested in this work can be separated in four classes: Methods that use no information. Methods that use, for each intersection, information about the incoming approaches of that intersection. Methods that use, for each intersection, information about the incoming and outgoing approaches of that intersection. And finally, methods that use, for each intersection, information about the incoming approaches of that intersection and neighbouring intersections.

It is shown that methods that use no environment information, such as the Random controller, perform the worst, as is expected, followed by the static and Webster controllers and independent learners, that use the incoming approaches of a single intersection. The best performing controllers are the Max-Pressure controller and the MARLIN controller, that use, information about the outgoing approaches and neighboring intersections, respectively. Thus, results show that, controllers with higher observability can obtain better performance when compared to controllers with lower observability.

It is also worth noting that, while microsimulators are used to develop RL-based traffic controllers and provide a start to improve urban mobility, there are a multitude of factors, that mostly regard safety and explainability, that need to be taken into account when moving from a simulator to the real-world. For example, simulating vehicle collisions, different weather types, pedestrian crossing, sensor noise and parking lanes are factors that could heavily influence the traffic environment that are not being currently simulated in this work.

References

[1] Joseph M. Sussman. *Perspectives on Intelligent Transportation Systems (ITS)*. Springer, 2005.

[2] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[3] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *CoRR*, abs/1904.08117, 2019.

[4] F.V. Webster. Traffic signal settings. Technical Report 39, British road res. Lab., 1958.

[5] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pages 27–66. Springer, 2013.

[6] Mohammad Aslani, Mohammad Mesgari, and Marco Wiering. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies*, 85:732 – 752, 2017.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[8] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers: Methodology and Large-Scale Application on Downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150, 2013.

[9] Roger P. Roess, Elena S. Prassas, and William R. McShane. *Traffic Engineering*. Pearson/Prentice Hall, 2004.

[10] John Little, Mark Kelson, and Nathan Gartner. MAXBAND: A Program for Setting Signals on Arteries and Triangular Networks. 795:40–46.

[11] Malik Tubaishat, Yi Shang, and Hongchi Shi. Adaptive Traffic Light Control with Wireless Sensor Networks.

[12] Marco Wiering. *Multi-Agent Reinforcement Learning for Traffic Light Control.*

[13] Marco Wiering, Jelle Veenen, Jilles Vreeken, and Arne Koopman. Intelligent Traffic Light Control.

[14] Marco Wiering, Jilles Vreeken, J. Veenen, and Arne Koopman. *Simulation and Optimization of Traffic in a City*.

[15] As'ad Salkham and Vinny Cahill. Soilse: A decentralized approach to optimization of fluctuating urban traffic using Reinforcement Learning. pages 531–538.

[16] Mohamed Khamis and Walid Gomaa. Enhanced Multiagent Multi-Objective Reinforcement Learning for Urban Traffic Light Control. 1:591.

[17] Thomas L. Thorpe and Charles W. Anderson. Traffic Light Control Using SARSA with Three State Representations.

[18] Xiaoke Zhou, Fei Zhu, Quan Liu, Yuchen Fu, and Wei Huang. A Sarsa(λ)-Based Control Model for Real-Time Traffic Light Coordination. 2014:e759097.

[19] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Design of Reinforcement Learning Parameters for Seamless Application of Adaptive Traffic Signal Control. 18.

[20] Eduardo Camponogara and Werner Kraus. Distributed Learning Agents in Urban Traffic Control. 2902:335.

[21] Wade Genders and Saiedeh Razavi. Using a Deep Reinforcement Learning Agent for Traffic Signal Control.

[22] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. *Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs*.

[23] Merlijn Steingrover, Roelant Schouten, Stefan Peelen, Emil Nijhuis, and Bram Bakker. *Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion.*

[24] Elise van der Pol. Deep Reinforcement Learning for Coordination in Traffic Light Control (MSc thesis).

[25] Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *ICML*, 1993.

[26] Pedro Pinto Santos. Trafic light control using deep reinforcement learning. Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, January 2021.

[27] Guilherme S. Varela, Pedro P. Santos, Alberto Sardinha, and Francisco S. Melo. A methodology for the development of rl-based adaptive traffic signal controllers. *arXiv:2101.09614*, 2021.

[28] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, 18(3):227–245, 2014.

[29] Pedro P. Santos, Guilherme S. Varela, Alberto Sardinha, and Francisco S. Melo. Reinforcement learning-based adaptive traffic signal control: A critical survey. *IEEE*, 2021.

[30] Gordon D. B. Cameron and Gordon I. D. Duncan. PARAMICS—Parallel microscopic simulation of road traffic. 10(1):25–53.

[31] Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, and Daniel Krajzewicz. *SUMO – Simulation of Urban MObility: An Overview*, volume 2011.

[32] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Jessie Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. *The World Wide Web Conference*, 2019.