IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 May 2023, accepted 2 June 2023, date of publication 6 June 2023, date of current version 13 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3283218

RESEARCH ARTICLE

A Reinforcement Learning-Based Distributed Control Scheme for Cooperative Intersection Traffic Control

JOSÉ A. GUZMÁN, GERMÁN PIZARRO, AND FELIPE NÚÑEZ, (Senior Member, IEEE)
Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Santiago 7820436, Chile

Corresponding author: Felipe Núñez (fenunez@uc.cl)

This work was supported in part by ANID under grants ANID BECA DE DOCTORADO NACIONAL CONICYT, AÑO ACADÉMICO 2019 21190519 and ANID PIA ACT210083.

ABSTRACT Traffic congestion is a major source of discomfort and economic losses in urban environments. Recently, the proliferation of traffic detectors and the advances in algorithms to efficiently process data have enabled taking a data-driven approach to mitigate congestion. In this context, this work proposes a reinforcement learning (RL) based distributed control scheme that exploits cooperation among intersections. Specifically, a RL controller is synthesized, which manipulates traffic signals using information from neighboring intersections in the form of an embedding obtained from a traffic prediction application. Simulation results using SUMO show that the proposed scheme outperforms classical techniques in terms of waiting time and other key performance indices.

INDEX TERMS Reinforcement learning, cyber-physical systems, intersection control, distributed control.

I. INTRODUCTION

Traffic congestion represents a serious problem for urban environments, due to its high monetary cost and the negative effects it has on the inhabitants of the area, caused by the generation of air pollution, delays in emergency services, and stress-related nervous diseases [1]. In 2020, there was a drastic decrease of vehicular traffic in large cities globally, caused mainly by the worldwide lockdowns imposed due to the COVID-19 pandemic. However, recent data shows that the upward trend in traffic congestion that was observed in previous years could quickly resume. Indeed, in 2021, traffic congestion in the United States costed drivers over $53 billion, a 41% increase from the previous year [2]. In this context, it is imperative to find efficient ways for reducing congestion.

The use of Urban Traffic Control (UTC) systems for efficient intersection management has been the standard solution to mitigate congestion. UTC systems are the backbone of modern Intelligent Transportation Systems (ITS) [3], which focus on the use of technology to effectively manage traffic [1]. Due to the extensive deployment of traffic detectors and the recent advances in computational equipment to efficiently process data, the current research trend in the area focuses on the use of data-driven algorithms to implement different high-level applications that help in improving traffic conditions [4]. In this setting, two of the most studied applications are: traffic prediction [5], which allows making strategic decisions to improve future traffic conditions; and automatic controllers that act in real-time on traffic signals, which allows taking immediate actions based on the current traffic conditions.

Due to the data-driven nature of the problem and the stochastic complexity, Neural Networks (NNs) have emerged as a natural tool to implement some of these applications, thanks to their ability to find complex relationships within the process [6]. Specifically, Reinforcement Learning (RL) NNs models are a promising choice for controller design, given their ability to learn optimal action policies from the interaction with the environment.

RL algorithms can be used to handle specific events, as in [7] where a transfer learning method is used to shorten the time needed to adapt to unexpected emergency situations, or to manipulate traffic lights in order to optimize traffic.

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

Some examples in this latter area are: i) the work in [8], which uses a Q-network to act on the traffic light, in conjunction with the Webster's method to achieve the stable learning performance of the agent; ii) the work in [9], which goes further by implementing a cooperative method based on the Proximal Policy Optimization (PPO), using a Convolutional Neural Network (CNN) as Actor-Critic, to decide the phase change moments of the traffic light; and iii) the work in [10], where, exploiting the idea of modeling traffic networks as graphs, a Graph Neural Network (GNN) is implemented as Actor-Critic model to learn cooperative control policies for phase changes in traffic lights, using the communication properties of GNNs.

Inspired by the latest developments and success of RL in areas as diverse as prognosis [11], social networking [12], and bio-inspired flocking control [13], in this paper we propose a RL-based distributed control scheme for cooperative intersection control. The proposed approach determines the state of the intersection by combining real-time data from traffic detectors at the intersection, with past and future traffic information, from the ego and neighboring intersections, encoded by a GNN traffic prediction model. This state serves as the input of a RL controller that uses a GNN to calculate, cooperatively, the ''split'' of traffic lights. Concretely, the main contribution of this work is the design of a distributed PPO RL controller for cooperative intersection control, which works in tandem with a GNN traffic prediction model that encodes the information of intersections and feeds the PPO RL controller. To illustrate the benefits of the proposed approach, an extensive simulation study is performed, which includes a comparison with classical approaches.

The rest of this manuscript is organized as follows. In Section II a brief review of related work on data-driven traffic prediction and RL-based controller design is presented. In Section III, the architecture of the proposed solution is detailed, describing the intersection model, traffic data and control actions used. In Section IV the GNN-based prediction model used to create the state embedding is introduced, while in Section V the RL controller is exposed in detail. In Section VI an extensive performance evaluation of the RL controller is given and, finally, in Section VII conclusions and directions for future work are stated.

II. RELATED WORK

The pertinent state of the art in the area includes the design of deep learning (DL) algorithms, which exploit the data collected from the transportation infrastructure, for the implementation of applications such as: traffic prediction or real-time controllers that manipulate traffic lights.

Regarding traffic prediction, in [14], a GNN traffic prediction model based on the Wavenet model for raw audio generation is presented. This model, called Graph WaveNet (GWNet), uses stacked blocks of dilated 1D convolutions and Graph CNNs to capture the hidden temporal and spatial dependency of the data. Training and testing of the model is done using the Highway of Los Angeles County (METR-LA) and the Caltrans Performance Measurement System detectors in the Bay Area (PEMS-BAY) datasets. Results show that the GWNet outperforms state of the art prediction models for prediction horizons ranging from 15 to 60 minutes.

In [5], a cyber-physical system for real time traffic prediction was implemented using detectors from Las Vegas I-15 Freeway. This system implements an on-the-fly data validation and reconciliation module, and a series of state of the art DL models for traffic prediction. The work in [5] shows the feasibility of performing real time traffic prediction with the available technology deployed in major urban areas.

On the controller synthesis side, an extensive number of works based on RL exist. In [8], a long short-term memory (LSTM) based Q-network to decide the action of the traffic lights is designed. The saturation of the roads, calculated using Webster's method, serves as the representation of the state of the intersections (i.e., the input of the RL network). The control action amounts to choosing the next green phase, relying on Webster's method to calculate the split of the phase. This is done to achieve a stable learning of the RL agent. This approach outperforms the traditional Q-table, a lookup table used to calculate the maximum expected future rewards for action at each state.

In a different approach, [9] seeks to improve performance by implementing a cooperative solution to decide control actions. To this end, the state of an intersection is defined as a spatial matrix where each element represents a section of the routes entering and leaving an intersection, and its value indicates if cars are present in that section. Then, all the matrices are grouped into a tensor that, in addition to a matrix indicating which phase of each intersection is green, represents the input to a CNN actor-critic. In this case, the output of the network is the probability of switching to the next phase in a predefined cycle, and the control action is defined based on this probability. To stabilize the training, two approaches are taken: on the one hand, the imitation learning technique is used to pre-train the model by making it to imitate an expert controller that increases the rate of output cars from the intersection; and, on the other, a PPO algorithm is used to reduce the impact of policy collapse (divergence of the policy) caused by the multi-dimensional output. This model was tested on a 9 intersection SUMO (Simulation of Urban MObility) environment against a pair of fixed time models and a pair of Q-learning models, using queue size and waiting time as performance indices. The cooperative control model obtained the best results, followed closely by the fixed-time models implemented. Interestingly, Q-learning models presented the worst performance. The work in [9] shows that cooperative control helps in improving the performance of RL controllers but, since in this particular formulation information from all intersections in the system is necessary to make a decision, scalability problems may arise when implementing it in real time.

Considering the observation that CNNs cannot effectively extract the dynamic features of the traffic, such as cars directions, [10] proposes a decentralized solution based on

a GNN multi-agent advantage actor-critic method, called GraphLight, to act on traffic signals. Since communication between agents is involved when using a GNN as actor-critic, this approach implies coordinated actions from the agents. Specifically, the state of an agent at time "t" is a vector containing the current traffic values measured by the detectors of its controlled intersection, combined with the state of neighboring agents multiplied by a spatial discount factor. The control action in this case is the selection of the next green phase that will be active for a fixed period of time "d". Testing was done on a 25 intersection SUMO scenario. Results show that the proposed method outperforms state of the art methods in terms of multiple metrics.

III. PROPOSED CONTROL SCHEME

The proposed control scheme considers that each intersection I_i is associated with an agent a_i that is able to communicate with a set of neighboring agents N_i. Communication between agents enables cooperation since access to the state information of the neighborhood $V_i = a_i \cup N_i$ will be available to predict the traffic at intersection I_i, as well as to take the control actions. Each intersection is required to be fully instrumented to allow real time traffic data acquisition and control.

For traffic prediction, it is proposed to split the prediction model into a GNN encoder and a linear decoder, so that it would be possible to distribute the encoder part among the agents and the decoder part can be implemented in a cloud layer, in the spirit of the general architecture presented in [15]. In the proposed scheme, each agent communicates the embedding of the prediction encoder to its neighbors and to the cloud layer; hence, the cloud receives the encoded data from all agents to make the final prediction. This way, each agent can use its own traffic data together with its own and neighboring prediction embedding as inputs to its RL controller.

To take advantage of the predictor structure, a GNN is used for the RL controller and each agent receives the embedding of the controllers of agents in N_i to calculate the control action. Under this setup, control actions are calculated from neighbors' information without the need of querying the cloud. Figure 1 shows a general schematic of the architecture.

A. INTERSECTION MODEL

For controller design, the intersection model presented in [16] (Figure 2) is considered. In this model, the concepts of movements, phases and cycles are used. A *movement* m_j is a flow of cars in a specific direction, and a *phase* $p_k \in P$ is a set of movements that flow simultaneously, within the time interval that this phase is active. As it can be seen in Figure 2, there are eight possible movements ($j \in \{0, 1, \ldots, 7\}$), where the even: m_0, m_2, m_4 and m_6 are left turns, and the odd: m_1, m_3, m_5 and m_7 are through-right combinations. For safety and efficiency, the model defines eight phases ($k \in \{0, 1, \ldots, 7\}$) as pairs of movements that flow without interrupting each other. Only these combinations are allowed.

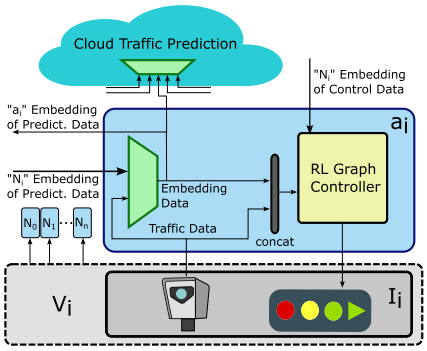


FIGURE 1. General architecture of the proposed RL-based distributed control scheme. At each intersection, local data, data from neighboring intersections, and high level traffic information from a cloud traffic predictor are used to produce the control action.

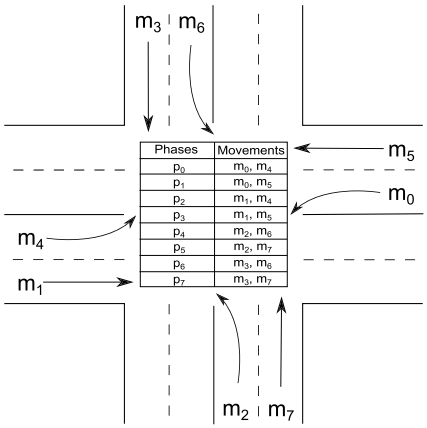


FIGURE 2. Intersection model consisting on eight *movements* that are grouped in eight *phases*, which contain a pair of *movements* that flow simultaneously.

Finally, a cycle C is a defined sequence of phases that is implemented in the traffic light in a cyclic manner.

B. INSTRUMENTATION AT THE INTERSECTION

It is assumed that each intersection is properly instrumented. Specifically, each intersection is equipped with detectors capable of collecting in real time different traffic variables from a specific area of the incoming roads. The detectors can be, for example, cameras pointing in the direction of the incoming roads that can extract traffic information using image processing. This way, the sensed area is bounded by the range of the camera. The traffic variables that are assumed available are:

- Vehicle density: Total number of vehicles in the sensed area divided by the approximate number of vehicles that can fit in this area.
- Vehicle queue: Total number of halted vehicles in the sensed area divided by the approximate number of vehicles that can fit in this area.
- Occupancy: Percentage of sensed area occupied by vehicles.
- Mean speed: Mean speed of vehicles in the sensed area.

Table 1 describes each of these variables, where max speed = 13.69 m/s. It should be noted that these measured

TABLE 1. Characteristics of measured variables.

Variable	Universe	Measure Unit
vehicles density	0 - 1	None
vehicles queue	0 - 1	None
occupancy	0 - 100	percentage
mean speed	0 - max speed	m/s

variables are indeed typically available in instrumented traffic infrastructure [1], [5].

C. CONTROL ACTION

The controller is a local object of intersection I_i that configures the activation times t_{p_k} of the phases $p_k \in P_i$ given a fixed traffic light cycle C_i. For simplicity and to add robustness to detector failures that prevent the collection of data from an area, a fixed cycle length Tc_i must be predefined by a theoretical or practical method. Then, starting from Tc_i, for all the intersections, the controller calculates the optimal green time tG_{p_k} (also called split) for each phase p_k when the cycle starts. The transition times between phases are defined as three seconds for the yellow time, two seconds for the all red time and five seconds for the minimum split time.

IV. TRAFFIC PREDICTION MODEL

A key feature of the proposed distributed control scheme is the use of future traffic information in the representation of the state of each intersection. To this end, we use the embedding produced by a GNN traffic prediction model. Unlike CNNs, GNNs are usually less complex, have information of the traffic flow direction (reference), and are usually scalable and easier to distribute, since each node only needs to access its local information and that of a subset of neighboring nodes.

To have a better understanding of the proposed strategy, it is necessary to introduce how graphs and GNNs work. Graphs are a type of data structure composed of nodes and edges, in which nodes represent modules that have information and edges connect nodes that are related. The advantage of working with graphs is that the information of the relationship between nodes is explicitly used. Mathematically, a graph $\mathcal{G}$ is represented as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. Each node $v \in \mathcal{V}$ connects to its neighbors with an edge $e \in \mathcal{E}$, and both have features, x_v and x_e, that change dynamically over time. Consequently, at each time step t the graph $\mathcal{G}$ has a feature matrix $X(t)$ [17]. The general principle of the GNN is to learn a neighborhood embedding h_v by aggregating information from the node neighborhood N_v through a process of ''neural message passing'', in which vector messages are exchanged between nodes and updated using NNs [18]. This process can be expressed as:

$$o_{N_v} = \text{aggregate}(h_u{}^k, \forall u \in N_v) \quad (1)$$

$$h_v{}^k = \text{update}(h_v{}^{k-1}, o_{N_v}), \quad (2)$$

TABLE 2. Movements from neighbors affecting the movements of a nominal intersection.

Movement	Input Movement	Output Movement
0	[2, 3, 5] East	[3, 6] South
1	[1, 6, 7] West	[1, 4] East / [3, 6] South
2	[4, 5, 7] South	[0, 5] West
3	[0, 1, 3] North	[3, 6] South / [0, 5] West
4	[1, 6, 7] West	[2, 7] North
5	[2, 3, 5] East	[0, 5] West / [2, 7] North
6	[0, 1, 3] North	[1, 4] East
7	[4, 5, 7] South	[2, 7] North / [1, 4] East

where ''update'' and ''aggregate'' are arbitrary differentiable functions that can be represented by NNs, and o_{N_v} is the aggregated message from neighborhood N_v. Hence, the inference of a GNN is an iterative process, where at each iteration k the aggregate function takes the set of embeddings from the neighboring nodes N_v to create a message o_{N_v} based on the aggregated neighborhood information. Then, in the same iteration, the update function combines the message o_{N_v} with the previous embedding $h_v{}^{k-1}$ to generate the new embedding $h_v{}^k$. At $k = 0$, the initial embedding $h_v{}^0 = x_v$, i.e., the features of the node v. A common practice is to represent each iteration with a NN layer with its own parameters, so that it is only necessary to stake layers until the desired number of iterations is achieved. Note that, at each iteration, the embedding h_v will include information from more distant neighbors, adding one ''hop'' per layer, making it a natural option to implement cooperative strategies.

To implement the GNN model, a graph is defined to model the traffic network, where each detector linked to a movement ''m_i'' represents a node of the graph and the edges are given by the connections with neighboring movements based on the direction of car flow. Table 2 shows the input and output connection with the neighboring intersection movements.

In this work, we consider a two layer GNN as the traffic prediction model, which is depicted in Figure 3. Using this particular structure is motivated by the results documented in [5]. To create the message o_{N_v}, the *aggregate* function is the element-wise multiplication between neighbors nodes features and connecting edges features, the latter represented by GNN parameters. The *update* function consists of passing the previous embedding through a feed-forward (FF) network, and then concatenating the result with the aggregated message o_{N_v}. The embedding h_v generated after the second layer is used as part of the input of the RL controller. Finally, two linear layers are implemented to generate the traffic prediction.

V. REINFORCEMENT LEARNING CONTROLLER

The proposed controller is a neural agent in actor-critic architecture based on PPO [19]. The fundamentals are exposed in the following.

A. PPO ALGORITHM

PPO is an on-policy RL algorithm that optimizes a surrogate objective and maximizes the expected advantage.

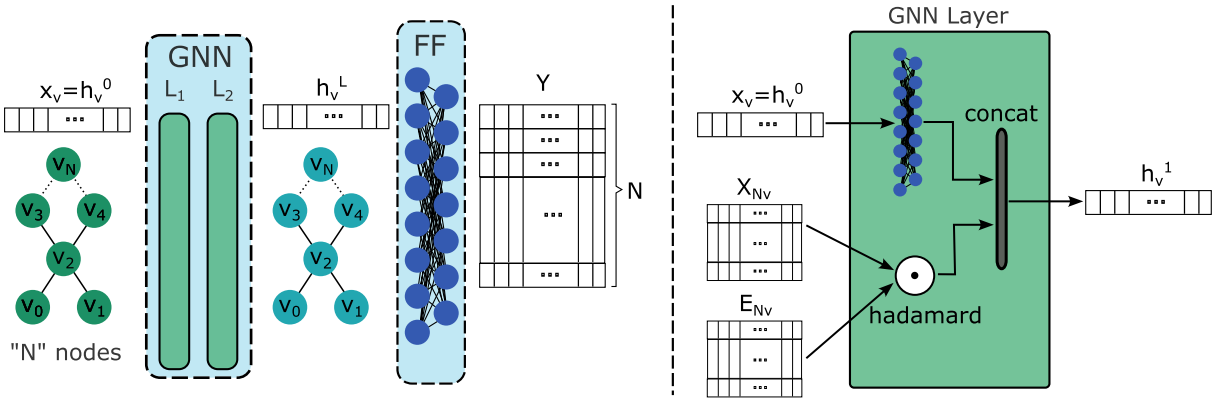


FIGURE 3. Architecture of the GNN network used for traffic prediction and generating the embedding used as input of the RL controller.

It is developed from the Trust Region Policy Optimization (TRPO) algorithm [20], where the objective is to find the biggest possible improvement step without stepping so far that accidentally causes a performance collapse. When this is guaranteed, it is possible to perform multiple iterations over the loss using the same trajectories, without leading to destructively large policy updates. TRPO determines the biggest step sizes using a second-order method, while PPO uses first-order methods that rely on approximations to keep the new policies close to the old ones. There are two variations of PPO, in this work we use PPO-Clip.

To give the context of PPO, the vanilla Policy Gradient (PG) and TRPO are introduced. The PG method works by optimizing the following loss,

$$L^{PG}(\theta) = \hat{\mathbb{E}}[\log \pi_\theta(a_t|s_t)\hat{A}_t], \tag{3}$$

where π_θ is a stochastic policy parametrized by θ, $\pi_\theta(a_t|s_t)$ is the probability of taking the action a_t given the state s_t and $\hat{A}_t$ is an estimator of the advantage function at timestep t. The expectation is taken as the empirical average over a finite batch of samples.

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, so $r(\theta_{old}) = 1$ and θ_{old} are the policy parameters before the update. TRPO maximizes the surrogate objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right], \tag{4}$$

subject to a constraint on the size of the policy updates based on the Kullback–Leibler (KL) divergence and guarantees monotonic improvement [20].

The superscript *CPI* refers to conservative policy iteration. Nonetheless, it uses second-order methods that are inefficient. Therefore, PPO modifies the objective by removing the constraint of TRPO and adding a penalization that prevents the policy from moving $r_t(\theta)$ away from 1. Consequently, PPO optimizes the surrogate objective [19]

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{5}$$

where ϵ is a hyperparameter used to keep the policy updates close to the old one. The motivation for this objective is as follows. The first term inside the min operator is the surrogate objective from TRPO, L^{CPI}. The second term clip$(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ is used to remove incentives for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$, which is equivalent to making small changes to the new policy π_θ with respect to the old policy. Finally, the minimum of the clipped and the unclipped objective is a lower bound on the unclipped objective.

As many other RL algorithms, PPO computes a variance-reduce advantage-function estimator making use of a learned state-value function $V_\phi(s)$ to compute the generalized advantage estimation (GAE). The main idea behind GAE is generating a bias-variance trade-off, which is based on the fact that trajectories close to the present generally have lower variance while variance increases considerably in future trajectories. A general way to present the advantages is through n-step returns given by

$$\hat{A}_t^n = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) + \gamma^n V_\phi(s_{t+n}) - V_\phi(s_t), \tag{6}$$

where $\gamma < 1$ is the discount factor, and the bigger n used, the higher the variance, while the lower n, the more bias. GAE generalize this advantage estimation by constructing all possible estimators and averaging them, namely,

$$\hat{A}_t^{GAE} = \sum_{n=1}^{\infty} w_n \hat{A}_t^n, \tag{7}$$

where $\hat{A}_t^n$ are weighted using an exponential falloff with $w_n \propto \lambda^{n-1}$ and $\lambda < 1$. When (7) is expanded, it simplifies to (8). In practice, GAE is computed by running the policy for T timesteps, and using the collected samples for an update. This is used to update an estimator that does not look beyond timestep T. Note that PPO's loss (5) is defined for any advantage estimation, but in practice uses

$$\hat{A}_t^{GAE} = \sum_{t'=t}^{T} (\gamma\lambda)^{t'-t}\delta_{t'} \tag{8}$$

Algorithm 1 PPO Algorithm

for iteration = 1,2, . . . **do**
  for actor = 1,2, . . ., N **do**
    Run policy π_{old} in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize L^{CLIP} wrt θ and L^{VF} wrt ϕ for K epochs. $L^{VF} = (V_\theta(s_t) - V_t^{targ})^2$.
  $\theta_{old} \leftarrow \theta$
end for

$$\delta_t = r(s_t, a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{9}$$

The PPO algorithm that uses a fixed-length trajectory is shown in Algorithm 1. Each iteration, there are N actors collecting T timesteps of data each. Then, (5) is constructed on these NT timesteps of data and optimized using Adam [21].

B. ACTOR-CRITIC MODELS

In the proposed control scheme, the actor and critic share structure with the traffic prediction model. Particularly, the actor π_θ and critic V_ϕ share a common feature extractor based on the traffic prediction model GNN. And then, each one projects the embedding vector using a two layer FF network to their respective outputs. The GNN feature extractor embeds the observation of each node (given in Table 1) and from their neighbors into a hidden vector h_v^L, where L is the number of GNN layers. After h_v^L is available at each node, each node computes the actions and the estimated expected reward given by π_θ and V_ϕ, respectively. It is worth noting that irrespective of the node, all use the same parameters θ and ϕ to compute the learned functions.

The actor outputs a vector $a_{v,t} \in \mathbb{R}^p$, where p is the maximum number of phases available on every intersection. $a_{v,t}$ is a probability vector, i.e., a real-valued vector with non-negative entries that sum 1, containing the proportion that each phase is green during the next traffic light cycle (if the intersection v does not have some phases, the entries of the action vector associated to the non-existing phases are masked). Meanwhile, the critic V_ϕ outputs a scalar containing the estimated expected reward for each node.

VI. IMPLEMENTATION AND EXPERIMENTATION

A. TEST BED IMPLEMENTATION

A scenario was designed for model testing in SUMO [22]. This scenario consists of a grid of nine intersections, where each intersection has eight incoming traffic lines and eight outgoing traffic lines. Each input line represents one of the movements of the model described in Section III, and has a traffic detector that covers an area equal to 50% of the road, i.e., partial observation of the road. In addition, each intersection has a traffic light with a fixed cycle C_i with the following sequence: $p_0 \rightarrow p_3 \rightarrow p_4 \rightarrow p_7$; the split time of each phase can be configured by the controller. Figure 4 shows the scenario implemented in SUMO.

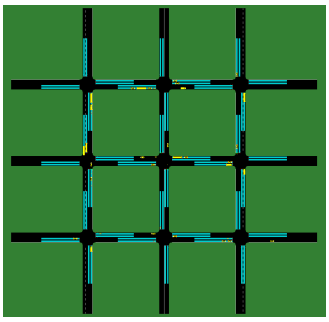


FIGURE 4. Scenario implemented in SUMO consisting on a grid with nine fully instrumented intersections.

The input data for the traffic prediction model is the last 12 samples of the variable ''mean speed'' that, with a sample rate of $5min/sample$, represent the mean speed during the past hour. The output are the next four samples, which represent the mean speed of the next 20 min. The embedding h_v for each node v is a vector of size 256. For training, 365 simulations, of roughly one day length each, were performed on the traffic network scenario. Each simulation was performed with a different traffic level, using a fixed-time controller for traffic light management and employing a normal distribution to choose the traffic level. Area detectors were used with a sample rate of 5 min to generate a dataset of the ''mean speed'' that represents one year of operation. Finally, the first 70% of the dataset was used for training, the next 10% for validation, and the last 20% for testing.

The RL controller was implemented using the Gym environment. Gym is a standard Application Programming Interface (API) for RL that implements a classic ''agent-environment loop'', where the *agent* makes an *observation* of the state of the environment, and then takes an *action* seeking to maximize a configured *reward* function. In our implementation, an agent is in charge of an intersection. The observation for an agent is an array $O \in R^{m \times s}$, with m equal to the number of input lines (movements) and s the length of the state vector of each line. The state vector is composed by the four traffic variables introduced in Section III, the embedding h_v delivered by the prediction model, and the present split value of the movement. The action $a_{v,t}$ is as introduced in Section V-B, and an observation followed by an action is taken each time a traffic light cycle starts. Several reward functions were evaluated, but the one that gives the best results is the ''vehicles difference'', defined as

$$rw = vl - vi, \tag{10}$$

where vl is the number of vehicles that left the line during green light and vi is the initial number of vehicles in the line before the green light.

Five traffic scenarios Si were created for training and testing the models. It was established that the models should be trained leaving a fixed traffic scenario, because

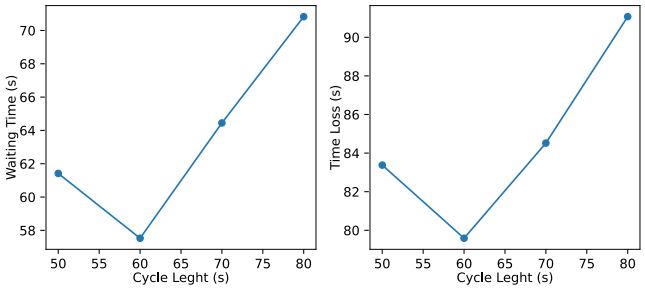


FIGURE 5. Cycle sensitivity test of the "RL Veh Left" controller evaluated in the traffic scenario S1 = 1540 vehicles per hour. The lower, the better.

traffic variations made the training unstable and an efficient controller was not reached.

To evaluate the proposed controller, two tests were conducted. First, a sensitivity test of the cycle length, where the RL controller was trained over the scenario S1 (1540 vehicles per hour) using different cycle lengths, so we can find the cycle length that gives the best performance over a fixed traffic scenario. And second, a performance test that seeks to evaluate the controller's capability to adapt to scenarios different from the one learned during training. In this test, five RL models were trained (one for each scenario) using the best cycle found in the first test, and then were tested against two classical controllers over all the designed traffic scenarios. RL models were named "RL Veh Diff Si", where "i" is the number of the scenario they were trained on.

All models were trained over 1000 epochs, using 28 trajectories for training and 14 trajectories for validation per epoch, where each trajectory consists of one hour of controller submission in the training scenario. Finally, the best-performing models over all epochs were stored for testing. In both tests, cars were introduced during the first hour of simulation, and the simulation finishes either when there are no cars left (all cars have reached their destiny), or after 7200 seconds in case the controller is not able to clear the cars.

The base codes for implementing the proposed approach are available in [23].

B. EXPERIMENTAL RESULTS

For both tests, as performance indices, we propose:

- Waiting time: time in which the cars had a speed lower than 1 m/s.
- Time Loss: Average time lost due to driving below the ideal speed (slowdowns due to intersections, etc.)

For testing the sensitivity to the cycle length, the controllers were trained and tested using five different cycle lengths, $Tc_i \in [40, 50, 60, 70, 80]$ seconds. Results are presented in Figure 5. For $Tc_i = 40$ seconds an efficient controller was never reached, so this result was not plotted. It can be seen that both performance indices find a minimum at $Tc_i = 60$ seconds and they deteriorate linearly both as this number increases or decreases. For this reason, a cycle length $Tc_i = 60$ seconds was used to train all the models in the following test.

For testing the performance, motivated by the results in [9] where a similar objective of cooperative intersection control with RL models is pursued, we compare the performance of our controllers with a fixed-time strategy, since this strategy was the closest competitor in [9]. This controller has a fixed cycle time, equal in length to the RL controller, but divided equally among the phases. To increase the meaningfulness of the test, it was decided to also compare against Webster's method, which is an optimal control strategy widely used in real implementations. A Webster's controller was designed for each traffic scenario *Si*, and to be fair over the test conditions, it was decided to constrain the cycle length of the Webster's controllers to 60 seconds, so only the split length of each phase was configured.

As mentioned before, models were tested over five different traffic scenarios. Each scenario has a different traffic density, but the traffic distribution is maintained. This was done to improve the stability of the training process. For evaluation purposes, each controller was run ten times on the same scenario, and the results reported in Table 3 correspond to the average. It can be observed that the *RL Veh Diff Si* controllers achieved the best performance in terms of both indices in the specific scenario they were trained on. It can also be seen that each controller outperforms the Webster's controller in the traffic scenarios. This demonstrates the capability of the RL controller to handle similar traffic density scenarios to those in which it was trained.

When the controllers face a scenario different from the one they were trained on, it can be seen that the performance of the RL controllers starts to deteriorate compared to the Webster's controllers. However, this behavior is slow and less significant when they move to lower congested scenarios, so it is possible to find a specific scenario where the RL controller performs better than Webster's in a wide traffic density spectrum. This is the case, for example, of the "RL Veh Left S3", which performs better than Webster's in all the tested scenarios. Figure 6 shows the described behavior.

To estimate the computational complexity and validate the feasibility of implementing the RL controller in real-time, an experiment was conducted to study the execution time required by the RL controller to calculate a control action. The test was conducted on a desktop computer running Linux using an Intel Core i7-7700 CPU @ 3.60GHz × 8 processor, an NVIDIA GeForce RTX 2070 graphics card and 32 GB of RAM. Figure 7 shows the results in the form of a histogram. It can be seen that most executions are close to 3 milliseconds, while the worst-case execution time is barely over 5.5 milliseconds. These results suggest that the RL controller can be used in real time as its execution time is much lower than the control period.

Finally, the impact of the number of parameters of the RL controller on the execution time is analyzed. To this end, two additional versions of the RL controller, with twice and four times the number of parameters of the nominal case, were

TABLE 3. Results of the evaluated controllers for different traffic scenarios.

Model	Vehicles per Hour				
	1320 (S0)	1540 (S1)	1769 (S2)	1980 (S3)	2200 (S4)
			Waiting Time		
Fixed Time	74.53	82.39	-	-	-
Webster	61.74	63.00	66.33	67.92	75.07
RL Veh Diff S0	56.73	64.55	72.40	81.88	106.66
RL Veh Left S1	61.01	57.53	67.21	71.75	79.76
RL Veh Left S2	62.10	62.98	63.11	71.93	81.78
RL Veh Left S3	60.40	60.60	65.68	65.93	74.77
RL Veh Left S4	63.18	63.94	68.43	69.11	72.90
			Time Loss		
Fixed Time	95.91	105.73	-	-	-
Webster	81.76	83.48	87.86	90.33	98.83
RL Veh Diff S0	76.26	85.59	94.86	106.54	133.97
RL Veh Left S1	81.01	77.74	88.93	95.10	104.65
RL Veh Left S2	82.14	83.73	84.38	95.32	106.93
RL Veh Left S3	80.11	80.92	87.31	88.50	98.78
RL Veh Left S4	83.54	84.83	90.42	92.15	96.83

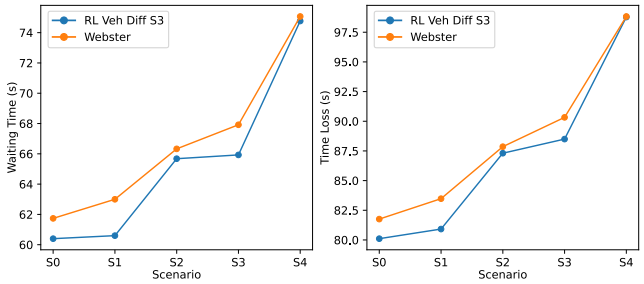


FIGURE 6. Comparison of waiting time and time loss indices between the RL Veh diff S3 controller and the webster controller. The lower, the better.

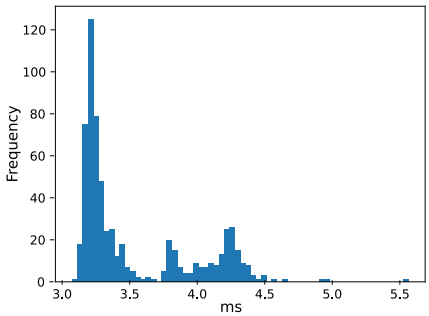


FIGURE 7. Histogram of the execution time of the proposed RL controller.

tested in terms of execution time. Figure 8 shows the results in the form of histograms. It can be seen that doubling the parameters has a mild impact on the execution time, while the controller with four times the parameters does present a significant increase of the execution time; however, it still presents a worst-case execution time much lower than the control period.

C. LIMITATIONS

Despite the promising results, some limitations exist. An important aspect of this work is that both training and testing were performed maintaining a constant input traffic distribution, as mentioned in Section VI-B. This

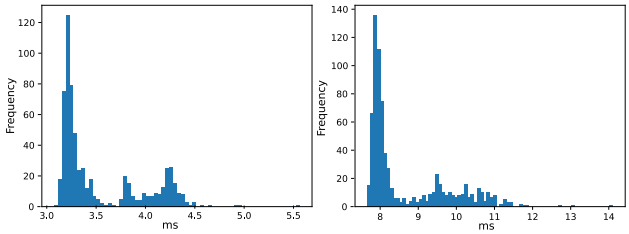


FIGURE 8. Histograms of the execution time of the RL controller when the number of parameters is varied. left: twice the parameters. right: four times the parameters.

means that, although several scenarios with different traffic densities were evaluated, the spatial distribution with which cars entered the network was almost always the same. This could be a limitation for the implementation in a real environment. A second limitation is that training and testing were performed in a scenario consisting of a grid of fully instrumented intersections of equal morphologies, which are very difficult to find in a real-world scenario. Further experiments are needed to evaluate and overcome these limitations.

VII. CONCLUSION

A novel RL-based distributed control scheme to address the intersection traffic control problem is proposed. The key ingredient is the use of an embedding from a GNN traffic prediction model to represent part of the intersection state, giving the solution a perspective of the future traffic level in the neighborhood. Simulation results obtained in SUMO show that the proposed controller outperforms the fixed-time and Webster's standard models for different traffic levels, even when traffic conditions are drastically different to those faced during training. It is also observed that the RL controller is more flexible to cycle time changes.

Future work includes increasing the complexity of the GNN models used for both traffic prediction and the RL controller, and using a model that allows extracting space-time information from the network, such as, for example, a distributed GWnet-based model.

REFERENCES

[1] J. A. Guzmán and F. Nuñez, "A cyber-physical systems approach to collaborative intersection management and control," *IEEE Access*, vol. 9, pp. 99617–99632, 2021.

[2] B. Pishue, "Inrix global traffic scorecard 2021," INRIX Res., WA, USA, Tech. Rep., 2021.

[3] X. Zhang and T. Riedel, "Urban traffic control: Present and the future," *Int. J. Urban Sci.*, vol. 21, no. sup1, pp. 87–100, Aug. 2017.

[4] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[5] J. A. Guzmán, B. T. Morris, and F. Nuñez, "A cyberphysical system for data-driven real-time traffic prediction on the Las Vegas I-15 freeway," *IEEE Intell. Transp. Syst. Mag.*, vol. 15, no. 1, pp. 23–35, Jan. 2023.

[6] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, "Deep learning on traffic prediction: Methods, analysis, and future directions," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4927–4943, Jun. 2022.

[7] Z. Mao, J. Li, N. Zheng, K. Tei, and S. Honiden, "Transfer learning method in reinforcement learning-based traffic signal control," in *Proc. IEEE 10th Global Conf. Consum. Electron. (GCCE)*, Oct. 2021, pp. 304–307.

[8] F. I. Shashi, S. M. Sultan, A. Khatun, T. Sultana, and T. Alam, "A study on deep reinforcement learning based traffic signal control for mitigating traffic congestion," in *Proc. IEEE 3rd Eurasia Conf. Biomed. Eng., Healthcare Sustainability (ECBIOS)*, May 2021, pp. 288–291.

[9] Y. Huo, Q. Tao, and J. Hu, "Cooperative control for multi-intersection traffic signal based on deep reinforcement learning and imitation learning," *IEEE Access*, vol. 8, pp. 199573–199585, 2020.

[10] Z. Zeng, "GraphLight: Graph-based reinforcement learning for traffic signal control," in *Proc. IEEE 6th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Apr. 2021, pp. 645–650.

[11] J. Lee and M. Mitici, "Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics," *Rel. Eng. Syst. Saf.*, vol. 230, Feb. 2023, Art. no. 108908.

[12] Z. Song, H. Guo, D. Jia, M. Perc, X. Li, and Z. Wang, "Reinforcement learning facilitates an optimal interaction intensity for cooperation," *Neurocomputing*, vol. 513, pp. 104–113, Nov. 2022.

[13] M. Jafari, H. Xu, and L. R. G. Carrillo, "A biologically-inspired reinforcement learning based intelligent distributed flocking control for multi-agent systems in presence of uncertain system and dynamic environment," *IFAC J. Syst. Control*, vol. 13, Sep. 2020, Art. no. 100096.

[14] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial–temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1907–1913.

[15] J. A. Guzmán, "A cyber-physical systems approach to collaborative intersection management and control," Ph.D. dissertation, Dept. Elect. Eng., Pontificia Univ. Católica de Chile, Chile, 2023.

[16] G. F. List and M. Cetin, "Modeling traffic signal control using Petri nets," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 3, pp. 177–187, Sep. 2004.

[17] Z. Liu and J. Zhou, *Introduction to Graph Neural Networks*, 1st ed. CA, USA: Morgan & Claypool, 2020.

[18] W. L. Hamilton, "Graph representation learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 14, no. 3, pp. 1–159, 2020.

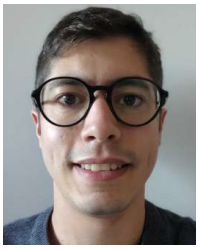
[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[20] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[22] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—Simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, pp. 128–138, Dec. 2012.

[23] G. Jose. (2023). *RL Distributed Control Scheme for Cooperative Intersection Traffic Control*. [Online]. Available: https://github.com/josegg05/rl_coop_traffic_control



JOSÉ A. GUZMÁN was born in Anzoátegui, Venezuela. He received the B.Sc. degree in electronic engineering from Universidad Simón Bolívar, in 2012, and the Ph.D. degree in engineering sciences from Pontificia Universidad Católica de Chile, in 2023. He is currently a Scientist with Pontificia Universidad Católica de Chile. His research interests include cyber-physical systems, intelligent transportation systems, urban traffic control, smart cities, and home automation systems.

GERMÁN PIZARRO was born in Santiago, Chile. He received the B.Sc. and M.Sc. degrees in electrical engineering from Pontificia Universidad Católica de Chile, in 2019 and 2021, respectively. His research interests include reinforcement learning and decision making, control theory, model predictive control, and distributed control.

FELIPE NÚÑEZ (Senior Member, IEEE) was born in Santiago, Chile. He received the B.Sc. and M.Sc. degrees in electrical engineering from Pontificia Universidad Católica de Chile, in 2007 and 2008, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara (UCSB), in 2014.

In 2015, he was with the UCSB/MIT/Caltech Institute for Collaborative Biotechnologies. In 2016, he joined the Department of Electrical Engineering, Pontificia Universidad Católica de Chile, where he is currently an Associate Professor. His research interests include networked control systems, cyber-physical systems, sensor and computer networks, industrial automation, distributed control, and mineral processing.

Dr. Núñez serves on the editorial board of *Control Engineering Practice*, an IFAC journal.