Towards Real-World Deployment of Reinforcement Learning for Traffic Signal Control

Arthur Müller, Vishal Rangras, Tobias Ferfers, Florian Hufen, Lukas Schreckenberg, Jürgen Jasperneite
Fraunhofer IOSB-INA
Lemgo, Germany
{givenname.surname}@iosb-ina.fraunhofer.de

Georg Schnittker, Michael Waldmann
Stührenberg GmbH
Detmold, Germany
{g.schnittker,m.waldmann}@stuehrenberg.de

Maxim Friesen
OWL University of Applied Sciences and Arts
Lemgo, Germany
maxim.friesen@th-owl.de

Marco Wiering
University of Groningen
Groningen, The Netherlands
m.a.wiering@rug.nl

***Abstract*—Sub-optimal control policies in intersection traffic signal controllers (TSC) contribute to congestion and lead to negative effects on human health and the environment. Reinforcement learning (RL) for traffic signal control is a promising approach to design better control policies and has attracted considerable research interest in recent years. However, most work done in this area used simplified simulation environments of traffic scenarios to train RL-based TSC. To deploy RL in real-world traffic systems, the gap between simplified simulation environments and real-world applications has to be closed. Therefore, we propose LemgoRL, a benchmark tool to train RL agents as TSC in a realistic simulation environment of Lemgo, a medium-sized town in Germany. In addition to the realistic simulation model, LemgoRL encompasses a traffic signal logic unit that ensures compliance with all regulatory and safety requirements. LemgoRL offers the same interface as the well-known OpenAI gym toolkit to enable easy deployment in existing research work. To demonstrate the functionality and applicability of LemgoRL, we train a state-of-the-art Deep RL algorithm on a CPU cluster utilizing a framework for distributed and parallel RL and compare its performance with other methods. Our benchmark tool drives the development of RL algorithms towards real-world applications.**

***Index Terms*—deep reinforcement learning, traffic signal control, intelligent transportation system, traffic simulation**

I. Introduction

Urban traffic is mainly controlled by intersection *traffic signal controllers* (TSC). Most TSC are actuator-based, i.e. they choose the traffic light signals according to the current traffic demand. Conventional TSC policies in most cases represent a set of many rules implemented by traffic engineers. Due to the dynamic and stochastic nature of traffic, manually designed TSC policies cannot control traffic optimally in all situations. This leads to inefficient use of existing infrastructure and thus to congestion, which has negative effects on the environment and human health [1]. It is also a huge economic problem. According to [2], the cost of traffic congestion in 2017 in urban areas of the United States is estimated at $179 billion.

This work is part of the KI4LSA project and was supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI).

Reinforcement Learning (RL), a long-standing research area in the field of artificial intelligence, is a promising approach to overcome the limitations of manually designed TSC policies. In RL an agent acting with an environment tries to optimize its behavior in order to achieve an optimization goal (e.g. smooth traffic flow). RL for traffic signal control has been extensively researched over the last two decades. In particular, the combination of deep learning and RL, referred to as Deep RL, has initiated an increased interest in RL for traffic signal control. This is because Deep RL has led to breakthroughs in solving complex problems such as the game of Go [3]. As a result of the extensive research made, there are several RL-based approaches in literature that have been shown to be superior to conventional TSC in simulation environments [4], [5]. Despite all efforts in this field, there is no real-world deployment of an actual RL-based TSC to date. In fact, it has not yet been proven that RL is even applicable as TSC in a real-world setting. Every investigation so far has been done only in simplified simulation environments [5]. To close the gap from training an RL agent in a simplified environment to deploy an RL-based TSC in the real-world, a realistic simulation environment is required as an intermediate step.

To this end, we introduce *LemgoRL*, a benchmark tool to train RL agents for traffic signal control in a realistic simulation scenario of an intersection in Lemgo, a medium-sized town in Germany. LemgoRL provides:

- a realistic model of a road network in Lemgo,
- simulation of all road user groups (vehicles, cyclists, and pedestrians) and not only considering vehicles,
- simulation of different vehicle classes (cars, motorcyclists, trucks, trucks with trailers, buses) and not only considering cars,
- simulation of the traffic volume according to measured data and not synthetic data,
- a set of realistic traffic phases designed by traffic engineers (domain experts),
- a mechanism to carry out transitions from one traffic light

phase to another in a realistic manner, taking into account safety and regulatory requirements,

- and a mechanism to ensure compliance with all safety and regulatory requirements (e.g. adherence to the minimum duration of traffic light phases) at all times during execution.

To the best of our knowledge, there is no simulation environment available in literature offering this degree of realism and complexity for RL-based TSC. LemgoRL is based on the programming language Python and offers the same interface like the well-known OpenAI Gym toolkit [6], allowing RL researchers easy integration into their work. It is an intermediate step in our research work towards the deployment of an RL-based TSC at a real-world intersection in the city of Lemgo. We provide LemgoRL as an open-source tool at https://github.com/rl-ina/lemgorl.

II. Preliminaries

A. Reinforcement Learning

In RL an *agent* is interacting with its *environment*. At each time step t, the agent gets information about the current situation of the environment called *state* s_t. Based on this information the agent selects the next *action* a_t to take in the environment. The behavior or decisions made by the agent are determined by the current *policy* π. It is a function that maps states to actions $\pi(a_t|s_t)$. Additionally, at each time step the agent receives a *reward* signal r_{t+1} from the environment, which is a scalar value. The goal of the agent is to maximize the cumulative reward in the long run.

Typically RL problems are formalized as *Markov decision processes* (MDPs). An MDP is a tuple consisting of the following elements [7]:

- A set of states $\mathcal{S}$,
- a set of actions $\mathcal{A}$,
- a state transition probability function $\mathcal{T}(s_t, a_t, s_{t+1})$ which defines a probability distribution over the successor state s_{t+1} given the former state s_t and action a_t,
- a reward function $\mathcal{R}(s_t, a_t)$ determining the reward r_{t+1}, when taking action a_t in state s_t,
- and a discount factor $\gamma \in [0, 1]$, discounting future rewards.

A central requirement for an MDP is the so-called Markov property, which assumes that the current state s_t contains all relevant information of the past states and actions. Thus, the successor state s_{t+1} only depends on the current state and action and not on the past states and actions. Furthermore, MDP models can represent *continuous*, meaning endless tasks, or *episodic* tasks in which there is a terminal state.

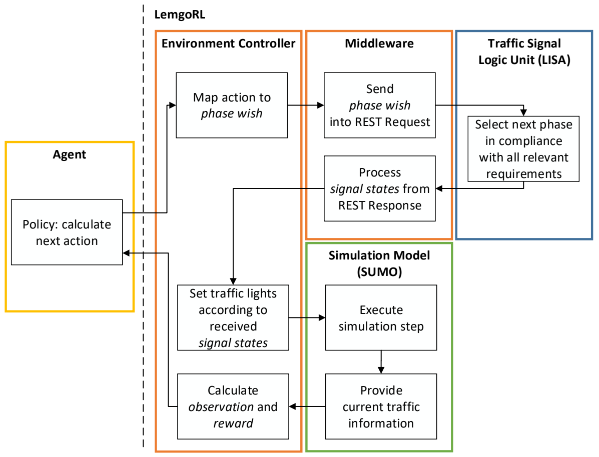
B. Traffic Simulator

In order to train RL agents for TSC, a simulation environment is needed. This is because, sampling data in reality would lead to sub-optimal decisions during training and therefore unacceptable waiting times for road users. RL algorithms are usually sample-inefficient, which would make training too slow. There are several traffic simulation tools that can provide the required simulation environment. According to [8], they can be categorized as macroscopic, mesoscopic, microscopic and submicroscopic simulators, with increasing levels of detail. For training and testing of TSC, usually microscopic traffic simulators are chosen which provide a reasonable tradeoff between simulation speed and required level of detail. Among these, there are commercial software tools like Vissim [9], Aimsun [10], and Quadstone Paramics [11] and free and open-source tools like MATSim [12] and SUMO [13].

For our simulation model, we use SUMO, mainly for two reasons. First, according to a recent survey paper [5], it is the most used tool for training RL as TSC. Over 70% of all analyzed papers about Deep RL approaches for TSC used SUMO in their work. Since we want LemgoRL to be highly applicable and easy to use for the research community, choosing a well-known and widely used traffic simulator helps achieving this goal. Second, SUMO provides an extensive interface to Python called TraCI[1] (Traffic Control Interface). This allows retrieving and setting values for most simulated objects during simulation runtime (e.g. speed of a vehicle), which is necessary to extract the relevant information for the MDP-setting.

III. Design of LemgoRL

LemgoRL consists of several software components including a simulation model made with SUMO, a traffic signal logic unit (TSLU) implemented with the traffic engineering software LISA, a middleware, and an environment controller, both implemented in Python. The environment controller encapsulates the components as a single Python class and provides an interface to the agent which is compatible with the OpenAI Gym toolkit.



LemgoRL
Environment Controller
Middleware
Traffic Signal Logic Unit (LISA)
Map action to phase wish
Send phase wish into REST Request
Select next phase in compliance with all relevant requirements
Agent
Process signal states from REST Response
Policy: calculate next action
Simulation Model (SUMO)
Set traffic lights according to received signal states
Execute simulation step
Calculate observation and reward
Provide current traffic information

Fig. 1: Process Diagram of LemgoRL

[1]https://sumo.dlr.de/docs/TraCI.html

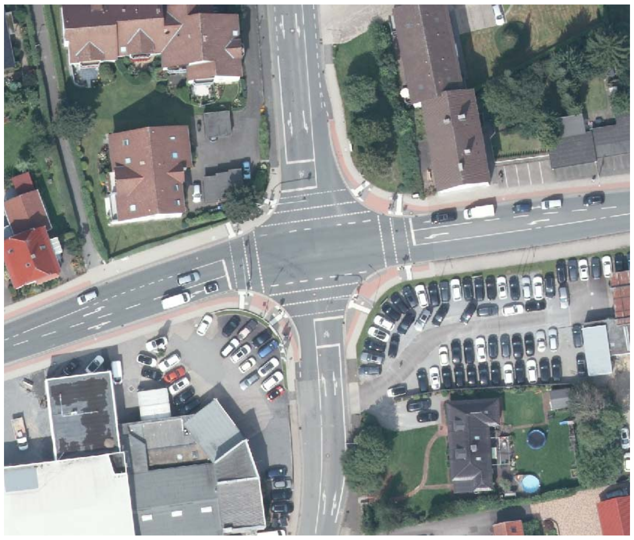
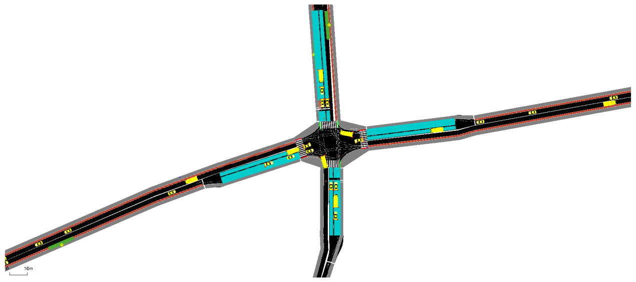
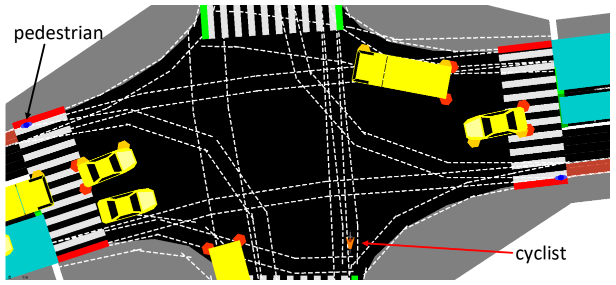


Fig. 2: Satellite Photo of the intersection "OWL322" in Lemgo



(a) OWL322 distant



(b) OWL322 close

Fig. 3: OWL322 intersection SUMO model

In Fig. 1 the process diagram of LemgoRL is shown. An RL agent computes the next action, which represents the agent's wish for the next traffic light phase. The environment controller passes this next action to the middleware, where it is packed into a message and sent via a REST API to the TSLU. There, it gets processed according to the logic designed by traffic engineers. Depending on the phase request and that logic, the TSLU keeps the current phase or initiates a transition to another one. Then a response message is sent back to the middleware containing the next phase with its corresponding signal states. The environment controller maps these signal states to the traffic lights in the simulation model. Following this approach, the simulated traffic lights are now in the traffic light phase, selected by the TSLU. Subsequently, SUMO executes a step in the simulation model and provides the information of the new traffic situation to the environment controller. The latter processes the information to obtain the state and reward signal of the underlying MDP and forwards them to the agent.

In the following subsections, the simulation model and the TSLU are explained in more detail.

A. *Simulation Model*

We built a simulation model for the intersection "OWL322" and its immediate vicinity in Lemgo (see Fig. 2). The intersection is known to suffer from congestion. The majority of traffic flows in the west-east direction. The traffic volume to and from the south is relatively low. All four incoming streets each have a shared lane for straight and right-turning traffic, as well as a separate left-turn lane. On the west and east street, bicycles share the road with vehicles. On the north and south street, bicycles have a separate lane with a dedicated waiting area in front of the vehicles. Furthermore, the intersection includes pedestrian crossings in all four directions and bus stations nearby, which are part of the simulation model as well. The simulation model is shown in Fig. 3. It consists of several files encoding information for the road network data, traffic demand, and traffic infrastructure, referred to as the *simulation scenario* in the SUMO setting [13].

The default way of communicating with a running SUMO instance is TraCI, which is based on a TCP client/server architecture. However, there is an alternative called Libsumo[2], which we also integrated into LemgoRL. It is a reimplementation of TraCI with static C++ functions avoiding the need for socket communication. Based on our observations, this brings an increase in performance of up to a factor of 10. However, the usage of Libsumo requires the additional effort of compiling SUMO from source. This is why we make it optional so that a user can choose between TraCI and Libsumo.

B. *Traffic Signal Logic Unit*

In order to have a realistic simulation environment, mechanisms are necessary to guarantee compliance with all safety and regulatory requirements. To this end, we developed a TSLU incorporating these mechanisms. The logic for these mechanisms was implemented by traffic engineers from our team, knowing and being able to apply all relevant requirements. The TSLU is based on the professional traffic engineering software LISA (Version 7.2)[3], so that the engineers were able to implement the logic in a domain-specific tool. LISA is a widely used and established commercial software package for planning, developing, and evaluating traffic signal controllers, that can be deployed in real-world traffic light systems.

[2]https://sumo.dlr.de/docs/Libsumo.html
[3]https://www.schlothauer.de/en/software-systems/lisa/

IV. Features of LemgoRL

In the following section, we want to highlight some features of LemgoRL.

A. Realistic Traffic Demand

In order to increase the realism in our simulation scenario we measured the actual traffic in the OWL322 intersection during the afternoon rush hour (from 15:00 to 16:10) and transferred the results to the model. We counted the number of cars, motorcyclists, trucks, trucks with trailers, buses, and bicyclists in 5-minute periods according to their respective directions of travel. The usage of these vehicle categories for measuring the traffic volume is recommended by the German Federal Ministry of Transport [14]. We also counted the number of pedestrians crossing the street in the same time periods. The total number of all road users over time is shown in Fig. 4.

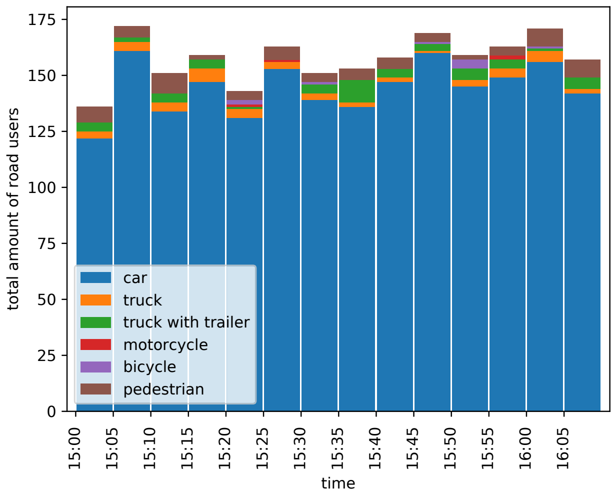


Fig. 4: Measured Traffic Volume

From the measured traffic data, we calculated the "spawning" probabilities for all vehicle types and pedestrians per second and integrated them in the model as SUMO "flows". The underlying probability distribution for a vehicle or pedestrian to spawn is the binomial distribution. By using these probabilities, each training iteration exhibits a slightly different traffic demand and dynamics. This provides the RL agent a richer dataset during training, potentially leading to more robust policies. We have open-sourced the measured data as well as a Python script to convert this data to SUMO flows along with LemgoRL. To further increase the realism of the model, we added all bus stops near the intersection and buses driving according to the routes and schedule of the local transport companies[4].

[4]https://teutoowl.de/owlv/, https://www.stadtbus-Lemgo.de/

B. MDP Specification

LemgoRL formalizes the problem of controlling the OWL322 traffic lights as an MDP in order to be solvable with RL algorithms. The MDP is episodic with one episode lasting $4200\,\mathrm{s}$, which corresponds to the recorded time interval of the traffic measurement. Unlike most other related work, we select a short interaction time of $1\,\mathrm{s}$ for interactions between the agent and its environment. This interaction time resembles the common processing time of traffic light systems in Germany, where the entire cycle from processing the detector values to setting the traffic light phases takes $1\,\mathrm{s}$.

1) State Definition: We define the state as

$$\begin{aligned} s_t = \{&queue_{t,l}, wave_{t,l}, avg\_speed_{t,l}, wait\_veh_{t,l},\\ &wait\_ped_{t,l}, cur\_phase\_id_t, des\_phase\_id_t,\\ &time\_elapsed\_curphase_t, time\_elapsed\_desphase_t, \}_{l\in L} \end{aligned} \tag{1}$$

where l denotes an incoming lane or pedestrian crosswalk of the OWL322. The definitions for $wave$ and $wait\_veh$ are taken from [15]. $wave$ is the number of nearing vehicles per lane, whereas $wait\_veh$ represents the waiting time of a lane's leading vehicle. $queue$ measures the length of all waiting vehicles in a lane (queue length), while avg_speed is the average speed of all vehicles in the respective lane. The waiting time for the first person waiting for a green signal at a crosswalk is given by $wait\_ped$. Furthermore, we also include a one-hot encoded vector representing the current phase cur_phase_id and one representing the phase desired by the agent des_phase_id. Additionally, we include the elapsed time in the current phase $time\_elapsed\_curphase_t$ and how long the agent has been desiring its phase $time\_elapsed\_desphase_t$.

We provide this rich state representation for two reasons. First, the proper state space design for TSC is still an open research question [5]. By providing this variety, researchers can select a subset as state space representations tailored to the requirements of the used algorithms. Second, the analysis conducted in [16] indicates that a broader scope of information in the state signal leads to better performance when facing non-stationarity due to changes in traffic dynamics. The latter is the case in real-world scenarios like LemgoRL.

2) Action Definition: Modern traffic light systems are usually controlled by a phase-oriented approach, i.e. the controller selects traffic light phases based on the current traffic situation or according to a fixed time scheme. A traffic light phase represents a combination of signal states (i.e. signal colors like red or green). It determines which non- or partially conflicting traffic flows are allowed to cross the intersection and which have to wait [17]. One phase for example would allow traffic on the west-east direction while stopping traffic on the north-south direction.

In LemgoRL, traffic engineers designed a set of 8 traffic light phases that are used in the current real-world implementation of the OWL322 (see table I). In phase 3, vehicles are allowed to travel on the west-east axis, while in phase 2,

TABLE I: Traffic Light Phases for OWL322

phase nr	veh west	veh east	veh north	veh south	ped w&e	ped n&s
1	r	r	r	r	r	r
2	g	g	r	r	r	g
3	g	g	r	r	r	r
4	g	r	r	r	r	r
5	g+left g	r	r	r	r	r
6	r	r	g	g	g	r
7	r	r	g	g	r	r
8	r	r	g	r	r	r

* vehicle traffic signal encompasses all directions (e.g. veh west=all vehicles coming from west and traveling to the north, east or south)
** r=red, g=green, left g=left turning vehicles have priority green

pedestrians are additionally allowed to cross the street in the north and south. Equivalently, north-south traffic is allowed for vehicles in phase 6, with pedestrians additionally allowed to cross the streets in the west and east in phase 7. Phase 4 and 8 only allow traffic coming from the west and north, respectively, reflecting the high traffic demand from these directions. Noteworthy is also phase 5 with a priority green signal for left turning traffic coming from west. It can only be reached when transitioning from phase 4 after waiting for several seconds to allow traffic (which came from the east) to clear the intersection. This phase handles the large traffic demand coming from the west and heading north. Phase 1 represents the "all-red-phase". It is required as an intermediate phase to ensure safe transitions, e.g. when transitioning from phase 3 (vehicles west-east) to phase 7 (vehicles north-south). Except for phase 1, all phases are accessible to the agent, so the action space comprises 7 discrete actions.

3) Reward Function: For the reward, we extend the definition in [15] by including the waiting time for pedestrians $wait\_ped$:

$$r_{t+1} = -\sum_{l\in L}(queue_{t+1,l} + \alpha_{veh}\cdot wait\_veh_{t+1,l} + \alpha_{ped}\cdot wait\_ped_{t+1,l}) \quad (2)$$

By adjusting the tradeoff coefficients α_{veh} and α_{ped}, the agent can be influenced to prioritize pedestrians over vehicles or vice versa. This could be used, for example, in congested cities to provide incentives for people not to use cars.

C. Realistic Traffic Light Phase Transitions

Whenever the agent selects a phase other than the current one, the TSLU checks whether the transition is allowed or not. For example, if the current phase is 2 (vehicles west-east, pedestrians north and south), a phase transition is only permitted if the minimum green time for pedestrians crossing the street is not undercut. Otherwise, a phase transition would violate safety requirements, as pedestrians would not have sufficient time to safely cross the street. In this case, the TSLU would ignore the agent's phase wish and would stay in the current phase.

Designing proper phase transitions is a sophisticated traffic engineering task. It requires determining all transition and intergreen times for "all combinations of conflicting traffic flows" [17]. The intergreen times are composed of crossing, clearance and entering times. They in turn depend on the permissible speed, the road user groups and actual topology and geometrical dimensions of the intersection. In contrast to the related work, where simplified phase transitions are used, LemgoRL provides realistic phase transitions for all possible combinations.

D. Transferability to Real-world Application

The ultimate goal of our research work is to deploy an RL agent trained with our benchmark tool as real-world TSC at the OWL322 intersection. Therefore, we designed our benchmark tool in way that supports this goal. By providing the agent with the real traffic phases, ensuring proper phase transitions by the TSLU and ensuring compliance to all safety and regulatory requirements at all times, the trained agent in combination with the TSLU can be deployed at the real OWL322 intersection. In Fig. 5, the schema for a real-world deployment is shown. It should be noted, that the current sensors at the OWL322 intersection are not sufficient to capture all necessary information for calculating the MDP state signal. Therefore, additional camera or radar sensors need to be installed.

The workflow of the real-world deployment is similar to that of LemgoRL. The raw data of the sensors will be processed in a feature extraction module running on an edge computer. It converts the data to MDP state signals. The state signal must be identical in dimensions with that from the training process in order to be applicable to the RL agent. Based on this state signal, the RL agent selects the next phase which will be handed over to the TSLU running on the field controller of the traffic light system. Finally, the TSLU sets the state signals for all traffic light devices of the intersection.

V. Evaluation

To demonstrate the functionality of LemgoRL we train an RL agent with our benchmark tool and compare its performance with baseline methods.

A. Experimental Setup

The traffic simulation settings follow the descriptions in Section IV. To extract all information needed for the state and reward of the MDP, we use SUMO's *laneAreaDetectors* on each incoming lane, which resemble camera sensors in real-world applications. The length of this detectors is $45\,\mathrm{m}$ in the west, east and north. In the south its length is $30\,\mathrm{m}$, since the road has a bend here. However, due to low traffic volume from the south, the laneAreaDetector is still able to track all incoming vehicles so that the state and reward do not get distorted.

We evaluate our RL agent and baseline policies over 20 test episodes. Each episode is generated with a different random

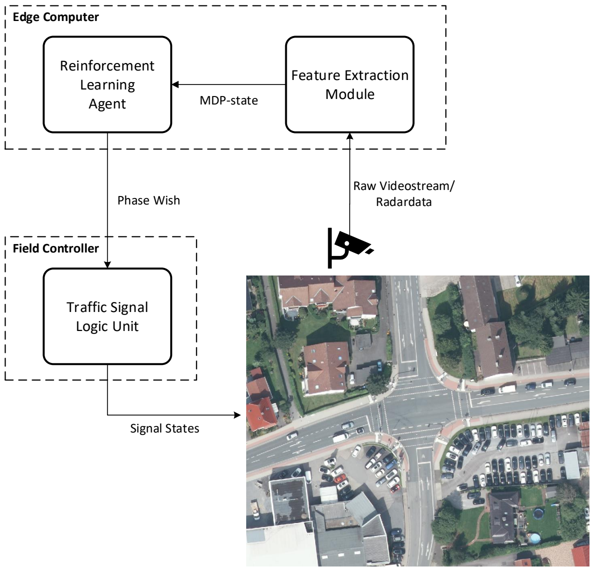

Edge Computer
Reinforcement Learning Agent
MDP-state
Feature Extraction Module
Phase Wish
Raw Videostream/ Radardata
Field Controller
Traffic Signal Logic Unit
Signal States

Fig. 5: Schema for Real-World Deployment of LemgoRL

seed, but the same random seeds are used for each policy to ensure a fair comparison. All methods are evaluated by the performance metrics average queue length, average waiting time for vehicles, and average waiting time for pedestrians, as these are the components of the reward function. Additionally, we add the average cumulative reward since this represents the overall performance of all three aforementioned metrics and the average speed.

B. RL and Baseline Policies

We use Proximal Policy Optimization (PPO) [18], which is a state-of-the-art deep reinforcement learning algorithm, to show the functionality of our benchmark tool. Since the performance of deep RL algorithms is sensitive to their hyperparameters, as shown in [19], we conduct a random search with 20 trials. Every agent is trained with 2000 episodes. The training is performed on a CPU cluster with 32 cores utilizing the RLlib framework [20] for distributed and parallel training on the cluster. As a result, the training time could be reduced drastically. Training the agent on a single core takes around $780\,\mathrm{min}$, while training on all 32 cores only around $90\,\mathrm{min}$. The hyperparameters of the best performing agent are shown in table II. For clarity reasons, the nomenclature follows that of RLlib. Another important aspect for successfully training an agent is normalization. Therefore we normalize the components in the state and reward function as shown in table II. After normalization, all values lie in the same range.

We compare the RL agent with three rule-based baselines to better assess its performance.

- *Fixed Timings Policy*: The traffic signals are switched in a fixed sequence, with the phase durations corresponding to that of the current control program for afternoon rush hour.

TABLE II: Hyperparameters used for PPO algorithm and MDP

Hyperparameter	Value
train_batch_size	8000
num_sgd_iter	10
gamma	0.98
lambda	0.95
vf_loss_coeff	0.1789
lr	$1.5e-5$
queue_norm	30
wave_norm	14
speed_norm	14
wait_veh_norm	14
wait_ped_norm	10
elapsed_norm	10
α_{veh}	1
α_{ped}	0.25

- *Random Timings Policy*: The traffic signals are switched in the same fixed sequence as in the Fixed Timings Policy, but the phase durations are sampled uniformly at random.
- *Greedy/Longest-queue-first Policy*: At each timestep, the traffic signal is selected that minimizes the immediate queue length.

C. Experimental Results

Fig. 6 and fig. 7 plot the vehicle related performance metrics, queue length and waiting time for vehicles, over the simulation time for all policies. The RL agent outperforms the baseline policies in both metrics. E.g., the average waiting time for vehicles is $37.9\,\mathrm{s}$ for PPO (see table III). Compared to greedy policy, which performs second best with $63.1\,\mathrm{s}$, this is a reduction by 39.9%.

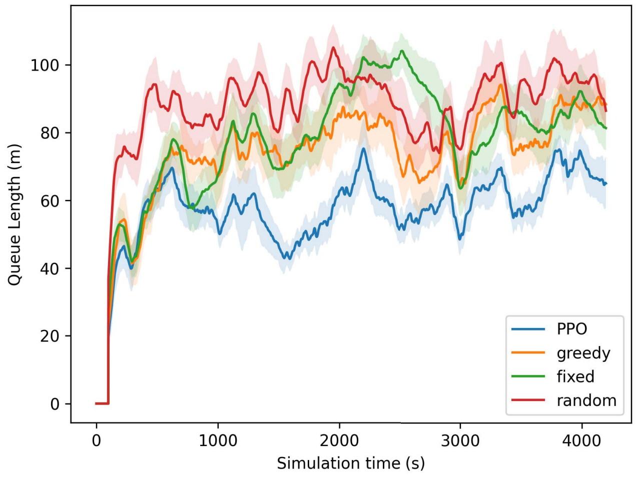


Fig. 6: Queue Length[5]

TABLE III: Performance Comparison. Best values are in bold.

Metrics	Random	Fixed	Greedy	PPO
cumulative reward	-5834.7	-3756.5	-3295.5	**-2269.7**
avg. queue length [m]	88.2	78.5	74.5	**57.9**
avg. waiting time veh. [s]	166.7	96.1	63.1	**37.9**
avg. waiting time ped. [s]	124.6	16.2	8.2	**2.0**
avg. vehicle speed [m/s]	7.5	**8.5**	7.3	7.8

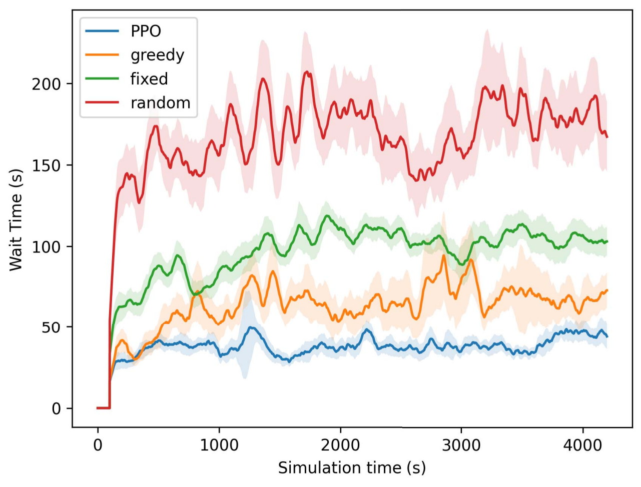


Fig. 7: Waiting Time for Vehicles[5]

When it comes to average waiting time for pedestrians, as shown in fig. 8, the RL agent significantly beats the other methods. In comparison to the greedy policy, PPO reduces the average waiting time for pedestrians by 75.6% and compared to fixed timings policy even by 87.7%. The main reason for this enormous difference is probable that the RL agent incorporates the minimization of the waiting time for pedestrians as optimization goal, while the rule-based policies focus only on vehicles. It is also noteworthy, that PPO performs better in both, vehicle and pedestrian related metrics, although these are opposing each others. This highlights the optimization potential that RL holds for TSC.

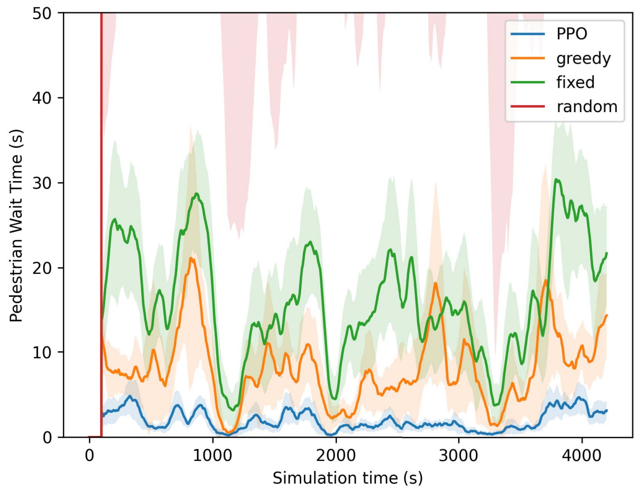


Fig. 8: Waiting Time for Pedestrians[5]

It is interesting to notice that the average vehicle speed is close to each other for all policies, even for the random timing policies, which performs significantly worse when it comes to average waiting time for vehicles and pedestrians. Fixed timings policy achieves the best results for average vehicle speed, performing 9% better than the RL agent.

[5]Shaded area represents standard deviation. To smooth-out the curves, we use a moving average with the window size of 100.

VI. Conclusion

In this paper, we have presented LemgoRL, an open-source benchmark tool to train RL agents as TSC in a realistic simulation environment. The experimental results demonstrate, that a state-of-the-art deep reinforcement learning algorithm like the PPO algorithm can be trained with our benchmark tool and outperform rule-based methods. Furthermore, LemgoRL was shown to work with a modern, high-performance RL framework such as RLlib, enabling parallelization so that e.g. training time can be drastically reduced.

By enabling researchers to train and evaluate algorithms in such a realistic environment and by providing a blueprint of how to ensure compliance with all regulatory and safety requirements, this tool drives the development of RL algorithms for traffic signal control towards real-world deployment. In future work, we plan to deploy agents trained with our benchmark tool on the real-world OWL322 intersection. In this way, we aim to prove the applicability and superiority of RL-based TSC in a real-world deployment.

Acknowledgment

The authors would like to thank Mirko Barthauer[6], Jakob Erdmann[7], Maximiliano Bottazi[7], and the company Schlothauer & Wauer for giving us helpful information about the interface to the LISA controller.

References

[1] S. Pandian, S. Gokhale, and A. K. Ghoshal, “Evaluating effects of traffic and vehicle characteristics on vehicular emissions near traffic intersections,” *Transportation Research Part D: Transport and Environment*, vol. 14, no. 3, pp. 180–196, may 2009.

[2] D. Schrank, B. Eisele, and T. Lomax, “Urban Mobility Report 2019,” The Texas A&M Transportation Institute, Tech. Rep., 2019. [Online]. Available: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, oct 2017.

[4] K. L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, “A survey on Reinforcement learning models and algorithms for traffic signal control,” *ACM Computing Surveys*, vol. 50, no. 3, 2017.

[5] A. Haydari and Y. Yilmaz, “Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–22, 2020.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” *arXiv preprint arXiv:1606.01540*, pp. 1–4, 2016.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[8] S. Krauß, “Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics,” Ph.D. dissertation, Universität zu Köln, 1998.

[9] M. Fellendorf, “VISSIM: A microscopic Simulation Tool to Evaluate Actuated Signal Control including Bus Priority,” *64th Institute of Transportation Engineers Annual Meeting*, pp. 1–9, 1994.

[10] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic Simulation with Aimsun," in *Fundamentals of Traffic Simulation*, 2010, pp. 173–232.

[11] G. D. B. Cameron and G. I. D. Duncan, "PARAMICS - Parallel microscopic simulation of road traffic," *The Journal of Supercomputing*, vol. 10, no. 1, pp. 25–53, 1996.

[12] A. Horni, A., Nagel, K., *The Multi-Agent Transport Simulation MATSim*. London: Ubiquity Press Ltd., 2016.

[13] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, P. Wagner, and E. Wiebner, "Microscopic Traffic Simulation using SUMO," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, pp. 2575–2582, 2018.

[14] "Richtlinien für die Straßenverkehrszählung 2020 im Jahre 2021 auf den Bundesfernstraßen," Bundesministerium für Verkehr und digitale Infrastruktur, Referat StB 11, Tech. Rep., 2021.

[15] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[16] L. N. Alegre, A. L. Bazzan, and B. C. da Silva, "Quantifying the impact of non-stationarity in reinforcement learning-based traffic signal control," *PeerJ Computer Science*, vol. 7, p. e575, 2021.

[17] Forschungsgesellschaft für Straßen- und Verkehrswesen, Ed., *Richtlinien für Lichtsignalanlagen : RiLSA : Lichtzeichenanlagen für den Straßenverkehr*. Cologne: FGSV Verlag GmbH, 2015.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[19] W. Genders and S. Razavi, "An Open-Source Framework for Adaptive Traffic Signal Control," 2019. [Online]. Available: http://arxiv.org/abs/1909.00395

[20] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," *35th International Conference on Machine Learning, ICML 2018*, vol. 7, pp. 4768–4780, 2018.

[6]Technical University of Braunschweig
[7]German Aerospace Center