RouteChoiceEnv: a Route Choice Library for Multiagent Reinforcement Learning

Luiz A. Thomasini
Graduate Program in Applied Computing,
Universidade do Vale do Rio dos Sinos
São Leopoldo, Brazil
luizalfredo@edu.unisinos.br

Lucas N. Alegre
Institute of Informatics,
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
lnalegre@inf.ufrgs.br

Gabriel de O. Ramos
Graduate Program in Applied Computing,
Universidade do Vale do Rio dos Sinos
São Leopoldo, Brazil
gdoramos@unisinos.br

Ana L. C. Bazzan
Institute of Informatics,
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
bazzan@inf.ufrgs.br

ABSTRACT

Multiagent Reinforcement Learning (MARL) has been successfully applied as a framework for solving distributed traffic optimization problems. Route choice is a challenging traffic problem in which driver agents must select routes that minimize their own travel times, taking into account the effect caused by other drivers. While MARL algorithms for route choice have been proposed, there is no library that provides a set of benchmarks and algorithms that can be used by researchers in the field. In this paper, we fill this gap by introducing Route Choice Env, a centralized library for MARL-based route choice research. It follows the PettingZoo API, which allows us to provide a standard set of environments and agents for reproducible experimentation. The library is publicly available at https://github.com/ramos-ai/route-choice-env.

KEYWORDS

Route Choice, Reinforcement Learning, Multiagent Systems, Benchmarking

1 INTRODUCTION

Reinforcement Learning (RL) [16] addresses sequential decision-making problems in which an agent learns to maximize a reward signal by interacting with its environment. In Multiagent Reinforcement Learning (MARL) [18], those problems become more difficult as many agents share the same environment and must learn to interact with each other. MARL environments have seen a rise in popularity as many systems become more intelligent and the techniques of learning algorithms become more sophisticated [3]. Tackling these types of problems is a challenging task, as much as designing the environment and designing the algorithms to interact with it.

Urban traffic engineering is one of the areas that have been approached with the MARL framework achieving promising results [2]. Among the various traffic problems tackled with MARL, in this work, we consider *route choice* in particular. In route choice, each driver must travel from an origin to a destination. To do so, drivers need to choose one out of a set of possible routes to take. From a system perspective, one may want to assign routes to drivers so as to minimize overall delays and travel times. The drivers, on the other hand, have their own preferences and operate selfishly to maximize their own objectives. Thus, in order to minimize congestion, the system's and the drivers' perspectives are equally important, though they usually conflict [15]. In fact, the literature has been concerned with both perspectives either independently or simultaneously [9].

Running experiments on traffic scenarios typically involves the use of traffic simulators. Popular traffic simulators like SUMO [5] and CityFlow [19] are efficient at simulating microscopic traffic dynamics. However, these simulators are highly specialized and difficult to set up for research, hindering their use for fast development and prototyping of MARL-based route choice algorithms. In the literature, previous work introduced libraries that provide sets of benchmark problems for RL and MARL research. The two most well-known such tools are the OpenAI Gym [3] (now named Gymnasium[1]), and the PettingZoo library [17]. Traffic-specific RL and MARL libraries have also been proposed, including CityFlow [19] itself, PyRL [12], and SUMO-RL [1]. Nonetheless, all these works focus on *microscopic* traffic simulation, which poses a huge burden on the simulation complexity, thus hampering the development of MARL algorithms.

In this work, we introduce RouteChoiceEnv, a new MARL library for the route choice problem. RouteChoiceEnv features a *macroscopic* simulator that models traffic dynamics by means of volume-delay functions, which are known to efficiently approximate real-world traffic conditions [2]. We developed a route choice environment for our simulator building upon the PettingZoo library, thus ensuring ease-of-use and facilitating the adherence of the MARL community. PettingZoo has many environments for training agents, but no environment is provided in the standard library or as a third-party environment to simulate the problem of route choice in traffic scenarios. Our efforts thus represent a first step towards route choice simulation as a Gym-like environment. We aim to provide the community with a standard library that allows for easy experimentation and extensibility in this context.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

[1] https://gymnasium.farama.org/

The main contributions of this work can be summarised as follows:

- A centralized repository for the community to use as a benchmark for the route choice problem in the MARL setting.
- A clean API that allows for the easy design of MARL route choice experiments.
- A public library and code base that is easy to extend and experiment with new agents and policies.

The rest of the paper is organized as follows. Section 2 overviews related work. Section 3 formalizes the route choice problem. Section 4 introduces the RouteChoiceEnv library. Section 5 presents experiments that evidence the correctness of the implemented algorithms. Finally, Section 6 discusses the final remarks.

2 RELATED WORK

In the context of MARL-based route choice, previous works have proposed methods for optimizing traffic flow and reducing congestion. Examples include a regret-minimizing MARL algorithm that converges to the Nash equilibrium [10], a policy gradient RL algorithm to define optimal tolls [4], and MARL algorithms that are guaranteed to converge to system-efficient equilibria [11, 13]. The interested reader is referred to [2, 6, 9] for a more comprehensive literature overview. Nonetheless, it becomes clear that the literature on MARL-based route choice fails to make results reproducible. This is a consequence of the lack of publicly available repositories collecting the source code of state-of-the-art algorithms and environments.

Other studies have focused on developing simulation environments and datasets for evaluating and comparing different RL algorithms. OpenAI Gym [3] introduced an API for defining reinforcement learning environments. It became the most used library among RL researchers. Recently, there has been a growing interest in developing standardized APIs for multiagent RL, such as PettingZoo [17]. The PettingZoo library provides a unified interface for defining multi-agent RL environments and interacting with them using a variety of RL algorithms.

In the context of RL-based traffic optimization, few works have proposed libraries for RL research. SUMO-RL [1] is a library that provides reinforcement learning environments for traffic signal control, which are built using the SUMO simulator [5]. In Zhang et al. [19], CityFlow is introduced as a novel optimized traffic simulator that can be used to instantiate MARL environments for traffic signal control. However, none of these libraries address the problem of route choice optimization.

To the best of our knowledge, there are currently no publicly available MARL repositories for route choice that follow a standardized API. In this paper, we introduce RouteChoiceEnv, a new MARL repository for route choice that is compatible with PettingZoo. Our repository library provides a set of benchmark environments and baseline methods for evaluating and comparing different MARL algorithms for route choice in transportation networks.

3 THE ROUTE CHOICE PROBLEM

The route choice problem consists in determining the best route for agents in a road network to take from an origin to a destination. We model this problem in a multiagent setting, meaning that all drivers in the network must learn which routes to take in order to reduce their travel costs.

For this work, we make certain simplifying assumptions regarding the route choice problem to make the problem more tractable. While we acknowledge that real-life drivers have access to real-time traffic information through modern navigation systems, our model focuses on agents who rely mainly on their experience. This assumption allows us to isolate the effects of individual learning on route choices and investigate the dynamics of the problem.

Although this model is simple, it is also quite general. Moreover, the metrics of optimal results are known in the literature. Then, these assumptions are reasonable as they build upon the foundational setting of routing games [14]. Furthermore, this simplified model can serve as a foundation for future research that incorporates additional real-world factors such as real-time traffic information, driver's preference, and departure time.

In this section, we introduce basic concepts related to route choice and present the modeling of the problem as a multiagent reinforcement learning (MARL) problem.

3.1 Road Network Modeling

A road network can be represented by a directed graph $G = (N, L)$, where nodes N represent intersections and links L represent the roads between intersections. Each link $l \in L$ has a cost $c_l : x_l \to \mathbb{R}^+$ associated with traversing it, which is affected by the flow x_l of vehicles on it. This implies that highly congested roads have a higher travel cost. The interested reader is referred to [9] for more details on how road networks and their elements are modeled.

A path from one node to another is called a route. A route is formally defined as a set of links between two nodes in the directed graph. We name these nodes as origin and destination. The cost of taking a route R is the sum of its link's cost, i.e., $C_R = \sum_{l \in R} c_l$. We refer to the minimum time required to complete a route as the free flow travel time. This is equivalent to the cost of a route when each of its links has flow $x_l = 0$. From a system perspective, the closer a route's cost is to its free flow travel time, the less congested that route is.

Let D be the set of drivers in the road network, and P be the set of origin-destination (OD) pairs. Each driver $d_k \in D$ is associated with an OD pair $P_i \in P$. We represent the association of a driver with an OD pair using a function $A : D \to P$ such that $A(d_k) = P_i$. Each driver poses a fixed flow (default is 1) into the road network. Note that routes in the system can have shared links. Hence drivers with different OD pairs can have an impact on each other's travel cost.

3.2 Problem Modeling

Reinforcement learning problems are typically modeled as Markov Decision Processes (MDPs). MDPs are composed of states, actions, a transition function, and a reward function. For this work, we modeled the environment as a stateless MDP as all actions are known a priori by the agents and we do not need to define state transitions.

As mentioned earlier, in our design, we are abstracting real-time traffic information, driver's preference, and departure time. Therefore, we only simulate the drivers' route choice and system's

flow dynamics. Incorporating these additional real-world features may be added to the model in future work.

In our model, each driver has an ID and a discrete action space N, which is the set of possible routes to take. The learning algorithms choose an action for each particular driver and the environment steps, assigning the flow of drivers to the network (thus simulating congestion as drivers are commuting to their destination). The drivers then experience the reward of the selected action (the travel time of the selected route). This process repeats for the next episodes.

All parameters of the road network are defined by their corresponding instances. This includes not only the network topology, but also the number of vehicles, the cost functions, etc. This is the common practice in the traffic engineering literature [9], as it offers enhanced reproducibility.

Although the problem was modeled as stateless MDP, finding a solution that both satisfies drivers' preferences and system equilibrium is not a trivial task due to the multiagent setting. In this setting, agents must adapt to the non-stationary environment dynamics. Next, we formally define the observations and rewards of our model.

Observation: In the proposed route choice problem, agents operate with minimal information about the environment. Therefore, system information such as the highest or lowest cost routes is not available to agents. No observation is provided beside the free flow travel time of routes. This information serves as a baseline reference for drivers when initially evaluating different routes. In our case, it was only used for setting initial values on the first episode of our algorithm.

To illustrate that, assume that each driver checks an online navigation service before leaving its origin but soon after loses connection to the online system. In that case, the driver would know the shortest route, but wouldn't be aware of real-time traffic updates.

Reward: The reward signal is the travel time of taking a route, which is the objective the agents want to optimize. The agents can transform the reward with a utility function that is best appropriate to its learning dynamics.

4 THE ROUTECHOICEENV LIBRARY

In this section, we detail how the library is structured and how it can be used for reproducible MARL experimentation.

4.1 Agent-Environment Interaction

By using the PettingZoo API, we were able to separate the environment from the learning agent. The interaction occurs when every agent chooses an action, and we step through the environment. In this phase, the environment assigns the flow of the drivers to the flow distribution matrix and calculates the travel time on every route to return as the reward signal for the agents.

We are using the Parallel API in our design, as all agents choose a route and commute at the same time. For future work, we could also implement different departure times for drivers using the AEC model from PettingZoo.

The following interaction can be seen in the code snippet below:

```
from route_choice_env.route_choice import RouteChoicePZ

# Initialize environment
env = RouteChoicePZ(net_name="OW")

for _ in range(ITERATIONS):

    # Query for action from each agent's policy
    action_n = {d_id: drivers[d_id].choose_action() for
         d_id in env.agents}

    # Step environment
    obs_n, reward_n, terminal_n, truncated_n, info_n =
         env.step(action_n)

    # Update strategy (Q table)
    for d_id in drivers.keys():
        drivers[d_id].update_strategy(obs_n[d_id],
             reward_n[d_id], info_n[d_id], alpha=ALPHA)

    # Update global learning rate (alpha)
    ALPHA = ALPHA * DECAY if ALPHA > MIN_ALPHA else
         MIN_ALPHA

    env.reset()
```

To create an instance of the environment we need to specify the network. It is also possible to specify how many vehicles each agent is controlling through the 'agent-vehicles-factor' parameter, so that each learning agent can control a higher number of vehicles. The idea of this parameter is to control the granularity of the simulation, so that it can be sped up by such factor.

The environment is composed of the following:

- A road network. Which in itself is composed of a set of OD pairs and, each OD pair, a set of routes.
- A set of drivers. Each is assigned to an OD pair and has a discrete set of actions to choose from (available routes for that OD pair).

Note that running this setting for N iterations requires the environment to reset at the end of the episode. This is necessary as the problem was modeled as a stateless MDP. After stepping through the environment every driver has reached its destination, so we need to reset in order to recreate the drivers.

RouteChoiceEnv allows the user to extract different information from the simulation. Here we define each of them and discuss how each can be used for research:

Free flow travel times: As discussed in Section 2, we expect each route to have a minimum cost for completing it. This may be used mainly as a baseline to acknowledge how much a road increased its cost due to congestion.

Flow distribution: This property is a special data structure in our library. Say that a network has $|P|$ origin-destination pairs and each OD pair has K possible routes to take. Our distribution then is a matrix of size $|P| \times K$. The flow of drivers is assigned to the matrix after the environment steps (after agents choose a route). It can be accessed at every episode of the experiment to display the distribution of drivers in the road network.

Also, the flow distribution can be used as intuition for performance. That If you are training learning agents, when the flow

distribution remains constant for a few episodes, it could mean that agents stopped exploring and found an optimal strategy to follow.

Another way one could use this property is to measure congestion patterns in the network topology. Aligning the flow distribution with the average travel time, one could infer that when the flow of drivers increases in route k_1, the average travel time is affected positively or negatively.

Average travel time: This important property from the road network can be defined as follows: let $|P|$ be the number of origin-destination pairs, and K_i be the number of routes for the i-th origin-destination pair. Let C_{ij} represent the cost of the j-th route for the i-th origin-destination pair, and F_{ij} represent the flow on this route. Then, the total cost TC can be calculated as:

$$TC = \sum_{i=1}^{P} \sum_{j=1}^{K_i} C_{ij} F_{ij}$$

Let F represent the total flow of the network. The average cost AC can be calculated as:

$$AC = \frac{TC}{F}$$

This property is the primary metric we are concerned with optimizing.

Routes cost sum: Let C_{ij}^t represent the cost of the j-th route for the i-th origin-destination pair at iteration t.

The routes costs sum, denoted as S_{ij}, is the accumulated sum of the costs for the j-th route of the i-th origin-destination pair up to the current iteration:

$$S_{ij} = \sum_{t=1}^{\text{iteration}} C_{ij}^t$$

Routes cost min: The routes costs min, denoted as M_i, is the minimum average cost for the i-th origin-destination pair up to the current iteration:

$$M_i = \frac{\min_{1 \le j \le K_i} S_{ij}}{\text{iteration}}$$

4.2 Networks

The RouteChoiceEnv library is compatible with Maslab's specification of road networks. As such, it can be tested with any instance available in the transportation networks repository[2]. The available networks include, but are not restricted to:

- B^1, .., B^7: the B^p network has $2p + 2$ nodes, $4p + 1$ link, 4200 drivers and a single origin-destination (OD) pair.
- BB^1, BB^3, BB^5, BB^7: expansions of the Braess graphs. The BB^p network has two OD pairs, $2p + 6$ nodes, $4p + 4$ links, and 4200 drivers.
- OW: network with 13 nodes, 48 links, 4 OD pairs, 1700 drivers, and overlapping routes.
- Sioux Falls: abstraction of the Sioux Falls city, USA, with $|N| = 24$ nodes, $|L| = 76$ links, 528 OD pairs, $d = 360{,}600$ drivers, and with highly overlapping routes.

[2]Available at https://github.com/maslab-ufrgs/transportation_networks.

4.3 Agents

In our library, each driver is controlled by an independent agent. An agent, in turn, implement an RL algorithm. Currently, the RouteChoiceEnv library provides the following agents:

- RMQ-learning. This agent implements the Regret-Minimising Q-learning algorithm [8], a tabular MARL algorithm that employs action regret as reinforcement signal. To do so, each agent keeps a history of experienced rewards for each action taken on every time step. After taking an action, the agent then estimates how much better or worse it performed, on average, up to time T for not taking only the best action regarding its experience. Agents using RMQ-learning are guaranteed to converge to Nash equilibrium in the limit.
- TQ-learning. This agent implements the Toll-based Q-learning algorithm [11]. Such an algorithm changes the reward definition by considering not only the travel time, but also the monetary expenses of the route. Such expenses are formalized by means of tolls. In particular, this algorithm employs the concept of marginal-cost tolls, which are computed a posteriori by each agent using only local information. This algorithm is known to align the equilibrium to the system-optimum, thus reducing the Price of Anarchy.

RouteChoiceEnv also enables different action selection schemes to be used. However, in its current form, only ϵ-greedy strategy is available.

4.4 Extending the Library

To facilitate extending the library, we provided a module that defines core interfaces. Inside the core module there is a DriverAgent class and a Policy class, both of which are abstract objects that define attributes and functions needed to implement new agents and policies to interact with the environment.

5 EXPERIMENTS

Before diving into the experiments and results, let us further explore and build some intuition on the route choice problem.

In our later experiments, we used the OW network. As described in section 4.2, there are 4 origin-destination pairs, 1700 drivers, and overlapping routes in this example scenario. Figure 1 depicts the available routes for an agent assigned to drive from node A to node L in the OW network. The set of 8 routes was made available through the parameter K mentioned later. This one agent is also competing with another 599 agents that are assigned to this OD, in addition to the other 1100 drivers that are assigned to other origin-destination pairs, but the links between nodes are overlapping. The interested reader may refer to Exercise 10.1 from Ortúzar & Willumsen [7] for more information on this experiment.

For simplicity, we are omitting important information on this setting. Other origin-destination pairs such as A|M, B|L, and B|L are not being displayed. Neither we are presenting the free flow of these routes or their links' cost function. Table 1 presents a summary of the results obtained in the OW network. From the table, one can build some intuition on what a good solution is. As seen, when using a random policy, route selection of agents remained too sparse. By contrast, when using a learning algorithm, the agents converged to more direct routes in order to avoid overlapping with agents from

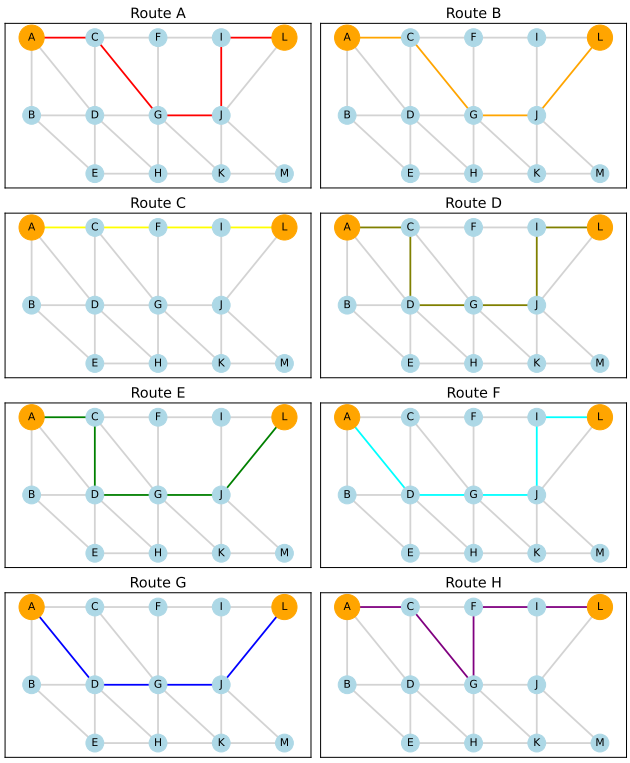

Figure 1: Set of 8 routes available in the RMQ-learning experiment for agents assigned to the A|L origin-destination pair.

different origins. Overall, this strategy has led to lower average travel time, which implies in less congestion over the network.

Then, in order to assess the correctness of our implementation, we performed some experiments to compare the results obtained by our implementation against the ones reported in the papers of the original algorithms [10, 11]. We ran experiments with both algorithms described in Section 4.3. Though these experiments are not intended to bring novel results, they still provide evidence of the correct implementations of the agents. As such, our experiments can be seen as benchmarks for the route choice problem in multiagent settings.

We selected the OW network and configured the RMQ-learning and TQ-learning agents with the same hyper-parameters reported in [10, 11], as presented in Table 2. We replicated each experiment 30 times. For each algorithm, we wanted to validate that our library can produce results that are consistent with the original ones.

Table 3 shows the results obtained by our and the original implementations of the algorithms. As can be observed, our results are consistent with the original ones. Nonetheless, we account that some differences can occur when reproducing experiments due to the setting and some randomness.

Additionally, we also plotted the average learning curves and standard deviation obtained by all algorithms in Figures 2 and 3. Again, as seen, the results show that the RouteChoiceEnv implementation of the algorithms is consistent with the original ones. We remark that we did not want to see an improvement in the learning

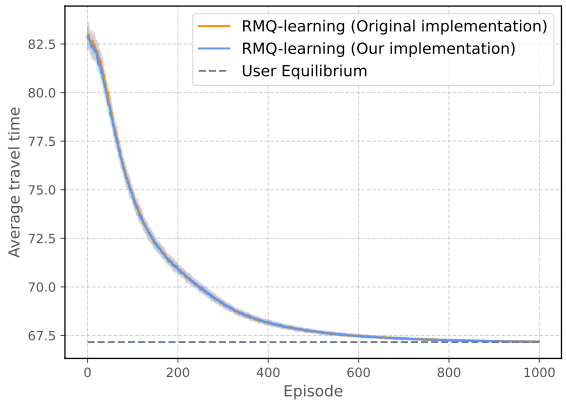

Figure 2: Comparison of the average learning curves obtained by the original and our implementations of RMQ-learning in the OW network.

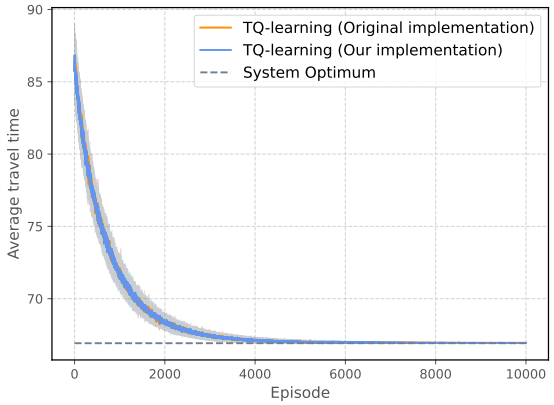

Figure 3: Comparison of the average learning curves obtained by the original and our implementations of TQ-learning in the OW network.

performance of our algorithms, but in the simplicity of running these experiments and to make it easily reproducible for anyone doing research in this area.

6 CONCLUDING REMARKS

This paper introduced RouteChoiceEnv, an extensible library for route choice simulation and experimentation with MARL algorithms. Our objective was to draw attention of the MARL community to this important problem as a benchmark for developing new algorithms.

In future work, we plan to implement other features to our simulator, such as net revenue and tolling schemes for the system, and real-life features, such as real-time information, departure times and drivers' preference (such as time and monetary preference).

Table 1: Flow distribution for over A|L routes and average travel time in the OW network. Reported results correspond to the last episode of experiments of a random policy versus RMQ-learning.

Algorithm	*Route A*	*Route B*	*Route C*	*Route D*	*Route E*	*Route F*	*Route G*	*Route H*	*Average travel-time*
Random policy	71	75	85	65	59	70	83	92	81.78
RMQ-learning	6	97	429	0	0	0	110	0	67.14

Table 2: Parameter configuration for each algorithm used when reproducing the experiments.

Algorithm	*Network*	*Episodes*	K	λ	μ
RMQ-learning	OW	1000	8	0.995	0.995
TQ-learning	OW	10000	12	0.999	0.999

Table 3: Comparison of the results obtained by our and the original implementations of the RMQ-learning and TQ-learning algorithms in the OW network. RMQ-learning results approximate the Nash equilibrium. TQ-learning results approximate the system optimum. In all cases, the closer to 1 the better.

Network	*Original Implementation*	*Our Implementation*
RMQ-learning	0.99975	0.99966
TQ-learning	0.99968	0.99965

Additionally, we plan to add novel networks and algorithms that the user can select for route choice experiments. Another interesting direction is to integrate our framework with other traffic simulators such as SUMO and CityFlow.

ACKNOWLEDGMENTS

This research was partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grants 140500/2021-9 and 304932/2021-3), Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - FAPERGS (grant 19/2551-0001277-2), and Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (grant 2020/05165-1).

REFERENCES

[1] Lucas N. Alegre. 2019. SUMO-RL. https://github.com/LucasAlegre/sumo-rl.

[2] Ana L. C. Bazzan and Franziska Klügl. 2013. *Introduction to Intelligent Systems in Traffic and Transportation.* Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 7. Morgan and Claypool. 1–137 pages. https://doi.org/10.2200/S00553ED1V01Y201312AIM025

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv preprint arXiv:1606.01540.

[4] Haipeng Chen, Bo An, Guni Sharon, Josiah Hanna, Peter Stone, Chunyan Miao, and Yeng Soh. 2018. DyETC: Dynamic Electronic Toll Collection for Traffic Congestion Alleviation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18).* AAAI Press, New Orleans, 757–765.

[5] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC).* IEEE, 2575–2582.

[6] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Behrouz Far, Zahra S H Abad, and Ana L. C. Bazzan. 2021. Reinforcement Learning in Urban Network Traffic Signal Control: A Systematic Literature Review. https://doi.org/10.31224/osf.io/ewxrj

[7] Juan de Dios Ortúzar and Luis G. Willumsen. 2011. *Modelling transport* (4 ed.). John Wiley & Sons, Chichester, UK.

[8] Gabriel de O Ramos, Bruno C da Silva, and Ana LC Bazzan. 2017. Learning to minimise regret in route choice. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems.* 846–855.

[9] Gabriel de Oliveira Ramos. 2018. *Regret Minimisation and System-Efficiency in Route Choice.* Ph.D. Dissertation. Universidade Federal do Rio Grande do Sul, Porto Alegre. http://hdl.handle.net/10183/178665

[10] Gabriel de O. Ramos, Bruno C. da Silva, and Ana L. C. Bazzan. 2017. Learning to Minimise Regret in Route Choice. In *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, S. Das, E. Durfee, K. Larson, and M. Winikoff (Eds.). IFAAMAS, São Paulo, 846–855. http://ifaamas.org/Proceedings/aamas2017/pdfs/p846.pdf

[11] Gabriel de O. Ramos, Bruno C. da Silva, Roxana Rădulescu, Ana L. C. Bazzan, and Ann Nowé. 2020. Toll-based reinforcement learning for efficient equilibria in route choice. *The Knowledge Engineering Review* 35 (2020), e8. https://doi.org/10.1017/S0269888920000119

[12] Gabriel de O. Ramos, Liza Lunardi Lemos, and Ana L. C. Bazzan. 2017. Developing a Python Reinforcement Learning Library for Traffic Simulation. In *Proceedings of the Adaptive Learning Agents Workshop 2017 (ALA2017) (ALA2017)*, T. Brys, A. Harutyunyan, P. Mannion, and K. Subramanian (Eds.). São Paulo. http://www.inf.ufrgs.br/maslab/pergamus/pubs/Ramos+2017ala.pdf

[13] Gabriel de O. Ramos, Roxana Rădulescu, Ann Nowé, and Anderson R. Tavares. 2020. Toll-Based Learning for Minimising Congestion under Heterogeneous Preferences. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, and G. Sukthankar (Eds.). IFAAMAS.

[14] Tim Roughgarden. 2007. Routing games. *Algorithmic game theory* 18 (2007), 459–484.

[15] Tim Roughgarden and Éva Tardos. 2002. How bad is selfish routing? *J. ACM* 49, 2 (2002), 236–259.

[16] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning: An introduction* (second ed.). The MIT Press.

[17] Justin K. Terry, Benjamin Black, Ananth Hari, Luis S. Santos, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, Caroline Horsch, and Praveen Ravi. 2020. PettingZoo: Gym for Multi-Agent Reinforcement Learning. *CoRR* abs/2009.14471 (2020). arXiv:2009.14471 https://arxiv.org/abs/2009.14471

[18] K. Tuyls and G. Weiss. 2012. Multiagent Learning: Basics, Challenges, and Prospects. *AI Magazine* 33, 3 (2012), 41–52.

[19] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference.* 3620–3624.