COMPUTER RESEARCH AND MODELING
2024 VOL. 16 NO. 1 P. 137–146
DOI: 10.20537/2076-7633-2024-16-1-137-146

UDC: 519.876.5, 519.179.2

Improving the quality of route generation in SUMO based on data from detectors using reinforcement learning

I. A. Salenek[1,a], Ya. A. Seliverstov[2,b], S. A. Seliverstov[2,c], E. A. Sofronova[3,d]

[1]St. Petersburg Institute of Informatics and Automation of the Russian Academy of Sciences,
39 14-th Line VO, St. Petersburg, 199178, Russia
[2]Solomenko Institute of Transport Problems of the Russian Academy of Sciences,
13 12-th Line VO, St. Petersburg, 199178, Russia
[3]Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences,
44/2 Vavilova st., Moscow, 119333, Russia

E-mail: [a] iasalenek@gmail.com, [b] seliverstov-yr@mail.ru, [c] seliverstov_s_a@mail.ru, [d] sofronova_ea@mail.ru

Received 21.11.2023.
Accepted for publication 21.12.2023.

This work provides a new approach for constructing high-precision routes based on data from transport detectors inside the SUMO traffic modeling package. Existing tools such as `flowrouter` and `routeSampler` have a number of disadvantages, such as the lack of interaction with the network in the process of building routes. Our `rlRouter` uses multi-agent reinforcement learning (MARL), where the agents are incoming lanes and the environment is the road network. By performing actions to launch vehicles, agents receive a reward for matching data from transport detectors. Parameter Sharing DQN with the LSTM backbone of the Q-function was used as an algorithm for multi-agent reinforcement learning.

Since the `rlRouter` is trained inside the SUMO simulation, it can restore routes better by taking into account the interaction of vehicles within the network with each other and with the network infrastructure. We have modeled diverse traffic situations on three different junctions in order to compare the performance of SUMO's routers with the `rlRouter`. We used Mean Absoluter Error (MAE) as the measure of the deviation from both cumulative detectors and routes data. The `rlRouter` achieved the highest compliance with the data from the detectors. We also found that by maximizing the reward for matching detectors, the resulting routes also get closer to the real ones. Despite the fact that the routes recovered using `rlRouter` are superior to the routes obtained using SUMO tools, they do not fully correspond to the real ones, due to the natural limitations of induction-loop detectors. To achieve more plausible routes, it is necessary to equip junctions with other types of transport counters, for example, camera detectors.

Keywords: transport modeling, multi-agent reinforcement learning, intelligent transport systems

Citation: *Computer Research and Modeling*, 2024, vol. 16, no. 1, pp. 137–146.

**КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2024 Т. 16 № 1 С. 137–146**
DOI: 10.20537/2076-7633-2024-16-1-137-146

Ки&М

СПЕЦИАЛЬНЫЙ ВЫПУСК

УДК: 519.876.5, 519.179.2

Повышение качества генерации маршрутов в SUMO на основе данных с детекторов с использованием обучения с подкреплением

И. А. Саленек[1,a], Я. А. Селиверстов[2,b], С. А. Селиверстов[2,c], Е. А. Софронова[3,d]

[1]Санкт-Петербургский институт информатики и автоматизации Российской академии наук,
Россия, 199178, г. Санкт-Петербург, 14-я линия ВО, д. 39
[2]Институт проблем транспорта им. Н. С. Соломенко Российской академии наук,
Россия, 199178, г. Санкт-Петербург, 12-я линия ВО, д. 13
[3]Федеральный исследовательский центр «Информатика и управление» Российской академии наук,
Россия, 119333, Москва, ул. Вавилова, д. 44, корп. 2

E-mail: [a] iasalenek@gmail.com, [b] seliverstov-yr@mail.ru, [c] seliverstov_s_a@mail.ru, [d] sofronova_ea@mail.ru

Получено 21.11.2023.
Принято к публикации 21.12.2023.

Данная работа предлагает новый подход к построению высокоточных маршрутов на основе данных от транспортных детекторов в пакете моделирования трафика SUMO. Существующие инструменты, такие как `flowrouter` и `routeSampler`, имеют ряд недостатков, таких как отсутствие взаимодействия с сетью в процессе построения маршрутов. Наш `rlRouter` использует мультиагентное обучение с подкреплением (MARL), где агенты — это входящие полосы движения, а окружающая среда — дорожная сеть. Добавляя в сеть транспортные средства с определенными маршрутами, агенты получают вознаграждение за сопоставление данных с детекторами транспорта. В качестве алгоритма мультиагентного обучения с подкреплением использовался DQN с разделением параметров между агентами и LSTM-слоем для обработки последовательных данных.

Поскольку `rlRouter` обучается внутри симуляции SUMO, он может лучше восстанавливать маршруты, принимая во внимание взаимодействие транспортных средств внутри сети друг с другом и с сетевой инфраструктурой. Мы смоделировали различные дорожные ситуации на трех разных перекрестках, чтобы сравнить производительность маршрутизаторов SUMO с `rlRouter`. Мы использовали среднюю абсолютную ошибку (MAE) в качестве меры отклонения кумулятивных данных детекторов и от данных маршрутов. `rlRouter` позволил добиться высокого соответствия данным с детекторов. Мы также обнаружили, что, максимизируя вознаграждение за соответствие детекторам, результирующие маршруты также становятся ближе к реальным. Несмотря на то, что маршруты, восстановленные с помощью `rlRouter`, превосходят маршруты, полученные с помощью инструментов SUMO, они не полностью соответствуют реальным из-за естественных ограничений петлевых детекторов. Чтобы обеспечить более правдоподобные маршруты, необходимо оборудовать перекрестки другими видами транспортных счетчиков, например, детекторами-камерами.

Ключевые слова: транспортное моделирование, мультиагентное обучение с подкреплением, интеллектуальные транспортные системы

Introduction

In the modern world, where the transport system of megacities is becoming increasingly congested and complex [Seliverstov et al., 2020a], accurate traffic modeling plays a key role in forecasting traffic flows, making it possible to estimate traffic volumes in different sections of the road network and predict its changes in the future [Seliverstov et al., 2020b]. Such information is necessary for infrastructure planning and the development of measures to reduce traffic jams and delays.

Accurate traffic simulation also allows one to optimize travel times. Knowing realistic data on traffic speeds and traffic jams on different sections of the road, it is possible to determine optimal routes and methods of movement, optimize the operating modes of traffic light control cycles, and thereby increase the capacity of the road network. This is especially important for urban and personal transport, where every minute matters.

Today, in the service of traffic management centers in megacities, intelligent traffic management systems are widely used as the basis of transport models. One such transport modeling software package is SUMO (Simulation of Urban MObility) [Santana, Sanchez-Medina, Rubio-Royo, 2015]. SUMO is an open source tool for transport system modeling and traffic research. It allows one to analyze various aspects of transport infrastructure, such as traffic flows, traffic signals, distribution of traffic flows by lanes, etc. [Lopez et al., 2018].

To run a full-fledged simulation in SUMO, one needs to build a road network scheme and set the routes of vehicles on it. Creating a schema is not a difficult task, due to the large amount of data available, such as mapping services with satellite images, passports with traffic light programs and other necessary information. This allows one to build high-precision schemes of the road network. On the contrary, building routes is a complex multimodal task, due to the lack of sufficient data for unambiguous calculation of routes. The main data for building routes are transport detectors that partially or completely cover the simulated network section. The most common types of detectors are point detectors (induction loop, radars) and area detectors (cameras). The former contain only information about the presence of a car at a given point at a specific time. The latter provide much more information, but are more complex in processing and storing data from them.

SUMO has several tools for building routes based on traffic count data [Behrisch, Erdmann, 2018]. Edge based counts are used in `dfrouter`, `flowrouter` and `routeSampler` tools. They use different algorithms to build routes based on data from detectors:

`dfrouter`

`dfrouter` requires that all inputs and outputs of the simulated network section be covered with detectors. The process of building routes consists of the following steps:

1. Separation of detectors in the network into 3 types: sink, source and between detectors.
2. Finding routes between detectors.
3. Calculation of the traffic flow between detectors. This step is performed under the assumption that the sums of the input and output detectors are the same for each measurement interval (thus the transit time is ignored). The number of vehicles generated at each step is determined only by the input detectors. Routes for the generated vehicles are distributed proportionally to the data from the output detectors.

`dfrouter` shows good results for sections of highways and motorways if all their entrances and exits are completely covered with detectors. In more complex urban networks, the routes generated may be implausible.

`flowrouter`

`flowrouter` is an improvement of the `dfrouter`. When building routes, it solves the problem of maximum flow, using data from detectors as a proxy for the throughput of network nodes. This allows the `flowrouter` to build more plausible routes. Also, the `flowrouter` copes better with omissions in data from detectors and incomplete coverage of a network section. It provides several options to limit the set of routes generated and thus allows one to calibrate the generated traffic.

`routeSampler`

`routeSampler` samples routes according to data from traffic counters. Sampling works iteratively by:

1. Selecting a random edge without overflow.
2. Choosing a random route through this edge that does not cause overflow in other edges.

Sampling occurs until all edges are filled, or until there are no available routes. The sampling probability can be set manually for each route. The shortage on the edges after sampling can be optimized by solving a corresponding Integer linear programming problem.

In many networks, the solutions for the given detectors' data are not unique. Thus, different SUMO algorithms can build completely different routes that correspond to the data from the detectors. This is due to the fact that the data from the detectors contain very little information about the actual routes of vehicles. One of the possible ways to utilize this information better is to take into account the interactions of vehicles with each other and with the network, which can be obtained from the simulation. This will make it possible to use the dependencies of detectors and traffic light objects, the time of passing edges, congestion, etc. Thus, we can expect a better match to the detector data, and possibly better routes.

None of the SUMO algorithms uses simulation directly in the process of building routes. Some of them can only be manually adjusted after simulating the routes they have built. To build routes based on SUMO simulation, we provide a new tool that we called `rlRouter`[1]. It utilizes a reinforcement learning algorithm, where agents (starting lanes) interact with environment (SUMO simulation) and performing actions (adding vehicles with possible routes) maximizing the reward (compliance with detectors data).

The rest of this document is organized as follows. In Section 2, we present a brief overview of the current literature on this problem. In Section 3, we provide a description of our `rlRouter` algorithm. Sections 4 and 5 contain experiments at simulated intersections and their results, respectively. Section 6 concludes.

Literature review

The paper [Zaky et al., 2017] discusses the use of SUMO to simulate traffic at two intersections. The method used to obtain flow arrival data is developed, the types of sensors used to obtain flow data are presented, the rationale for conducting simulations with actual conditions is presented, and recommendations are given on how to connect SUMO to MATLAB via TraCI. The parameters used in the simulation are shown to be critical as they become the threshold for how representative the simulation of real-world conditions is. Several parameters were used in the SUMO simulation: vehicle speed, acceleration, deceleration, distance between vehicles, and the size of each vehicle type. The simulation results confirm the ability of SUMO to simulate the behavior of two intersections.

[1] https://github.com/iasalenek/rlRouter

In the paper [Baumgart, Burger, 2021], methods for optimizing traffic flows using V2I technologies and optimizing traffic light control cycles using reinforcement learning were explored. The creation of a virtual transport environment and modeling of traffic scenarios was carried out in the SUMO package. A reinforcement learning approach was used to teach controllers (agents) to control specific vehicles and traffic light cycles. The paper shows how, using real-time information from other vehicles, the agent iteratively learned to improve traffic flow through repeated observations and algorithmic optimization. For the RL approach, various control policies have been considered, including widely used neural networks, as well as linear models and radial basis function networks. A comparison of an RL-trained controller with other control approaches is presented and the reliability of a traffic light controller in extreme scenarios is analyzed.

The work [Kheterpal et al., 2018] discusses the Flow framework, which allows integrating SUMO with the deep reinforcement learning libraries rllab and RLlib. The article shows that using these libraries makes it possible to apply deep reinforcement learning (RL) methods to traffic scenarios and obtain optimal traffic control plans in various traffic conditions. The work [Wu et al., 2021] discusses deep reinforcement learning (RL) methods applied to autonomous vehicle routing tasks. It presents a modular learning structure that uses deep RL to solve the problem of dynamic traffic flow routing. The modules are designed to handle different road traffic conditions (traffic jams, lane changes, intersection crossings). During the study, control laws for autonomous vehicle routing were identified, which help eliminate traffic congestion.

The work [Wesemeyer, Trumpold, 2019] presents the results of the Managing Automated Vehicles Enhances Network research project [MAVEN, 2023]. This project is aimed at developing a system for organizing traffic flow in the “green wave” mode using V2X infrastructure. V2X communication protocols are used to incorporate vehicles (ACVs) into a traffic simulation model at a real intersection. Until now, real traffic could be introduced into the simulation using stationary detectors, such as magnetic field sensors, induction loops, cameras, radars, etc. The disadvantage of this monitoring method is that only instantaneous information and, for example, behavior of vehicles approaching an intersection can only be estimated approximately. V2X-enabled vehicles constantly broadcast their position and speed through a cooperative data communication (CAV) system, allowing the model to accurately reflect vehicle dynamics. This study demonstrates the application of cooperative data exchange to optimize vehicle routing with V2X using traffic simulation in SUMO at a real Braunschweig junction.

The use of Multi-Agent Reinforcement Learning (MARL) for optimal taxi fleet management is explored in the scientific article [Liu et al., 2022]. The authors specifically focus on how electric vehicles learn to manage battery charging, picking up, and dropping off passengers.

In the article [Chouiekh et al., 2022], a strategy for controlling intersections by managing traffic signals is developed using Deep Reinforcement Learning (Deep RL) based on the Q-learning algorithm. The agent is controlled in a simulated road environment using the SUMO road simulator.

Methodology

`rlRouter` uses multi-agent reinforcement learning (MARL) to train the optimal vehicle routing policy. In this section we describe the key elements of the Markov Decision Process in relation to this task: the action space, the observation space, and the reward function. We also introduce our implementation of parameter sharing Deep Q-Network, which we used as the main RL algorithm.

Agent

Incoming lanes on the simulated section of the road network act as agents. Each step of the simulation, agents perform actions by adding a vehicle at the beginning of the corresponding lane or skipping this step. Each added vehicle is assigned one of the possible routes starting from the agent's lane. Thus, the action space dimension of each agent is:

$$\text{action}_{\text{dim}} = N_{\text{strarting routes}} + 1.$$

Environment

Interactions between agents occur within a SUMO simulation of a section of the road network. Each step of the simulation is performed in the following sequence:

1. The order of agents' actions at the current step is randomly selected (this avoids systematic shifts in the construction of routes).

2. For each agent, an observation vector consisting of the current state of the environment and the actions of previous agents is sequentially considered. Based on the observation vector obtained, the agent selects an action according to the current policy.

3. The environment applies agent actions and takes a simulation step.

There are several groups of observations in the observation space of each agent (Table 1). From traffic lights we include One-Hot encoded information about their current phase and its duration in seconds. To account for the vehicles in the network, we divide the network into sectors so that there can be no more than one vehicle in each sector. For each sector, we include One-Hot encoded information about the route of the vehicle located on it. From each detector we take information about their current shortage/overflow compared to ground truth data. We also include future ground truth observations from detectors for a certain period, for further processing using a recurrent neural network. Since the agents act in turn, we also include the actions of all other agents in the observation vector of each agent.

Table 1. Elements of the agent's observation space

Source	Observations
Traffic lights	One-Hot encoding of the current phase Duration of the current phase
Lanes' sectors	One-Hot encoded route of the vehicle located in this sector
Detectors	Current shortage/overflow compared to ground truth data Future observations for a given period
Agents	One-Hot encoded actions of other agents

Model

In this work, we use multi-agent DQN with parameter sharing model to train a cooperative policy of lanes-agents. The architecture of the Q-network is shown in Figure 1. It starts with a reversed recurrent neural network to process sequences of future observations from detectors. The output of the reversed LSTM it is concatenated with unit observations and passes through several fully connected layers. The first fully connected layers are separate for each agent, while the last layers are shared between agents. The Bellman equation for the DQN is:

$$Q^{\pi}(s, a) = R + \gamma Q^{\pi}(s', \pi(s')),$$

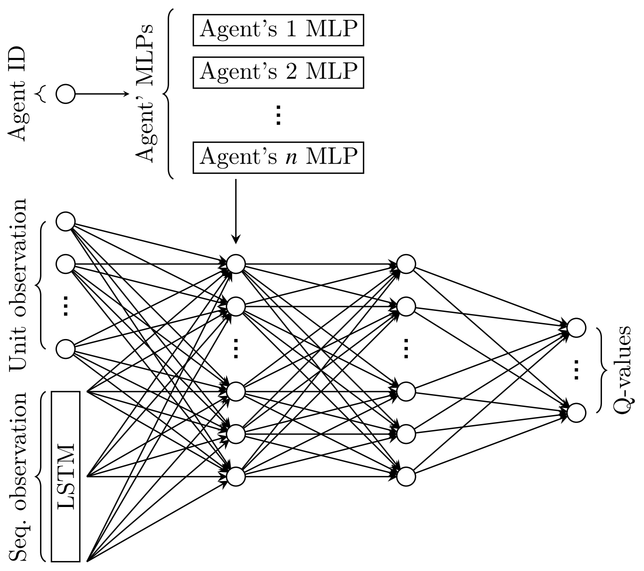

Figure 1. Q-network architecture. The reversed LSTM serves as a pre-processor for future sequential observations. Its output is concatenated with unit observations and passes through several fully connected layers. All fully connected layers except the last one are trained independently for each agent. The last layer is common to all agents

where as a reward we used $R = \sum_{d \in \mathcal{D}} r_d$, where

$$r_d = \begin{cases} 1, \text{ if vehicle detected and detector surplus } \leqslant 0, \\ -1, \text{ if vehicle detected and detector surplus } > 0, \\ 0, \text{ if vehicle was not detected} \end{cases}$$

and $\mathcal{D}$ is the set of detectors through which the routes of this agent pass.

Since a number of steps can pass from the moment the vehicle is added to the passage of all detectors, we are faced with the problem of delayed reward. To deal with it, we update the replay buffer with a new transition only when the added vehicle passes the last detector on its route.

Experiments

We conducted a series of experiments at simulated junctions (Figure 2) to compare SUMO routers (`flowrouter` and `routeSampler`) and our `rlRouter`. The first two junctions are the intersections of two one-way roads. At the first junction, the roads are single-lane, while at the second junction they are two-lane. The third junction is the intersection of 4 two-lane roads. All junctions are fully covered by induction-loop detectors located directly in front of the stop lines. There is one traffic light object at each junction, whose logic includes 4 phases: two main phases lasting 42 seconds and two intermediate phases lasting 3 seconds (total cycle time 90 seconds).

For each junction, we conducted a series of experiments with different flow values for each route. The value of the traffic flow was set as the probability of adding a vehicle following the route at each second of the simulation. This probability remained unchanged throughout the simulation, which was 3600 seconds (1 hour). To reveal the features of the work of different routers, for the first two junctions, we conducted 8 experiments, each differing in the ratios of incoming and outgoing flows, as well as their splits (Table 2). For the last junction we have provided only one experiment due to the large number of possible flow ratios and greater computational complexity.

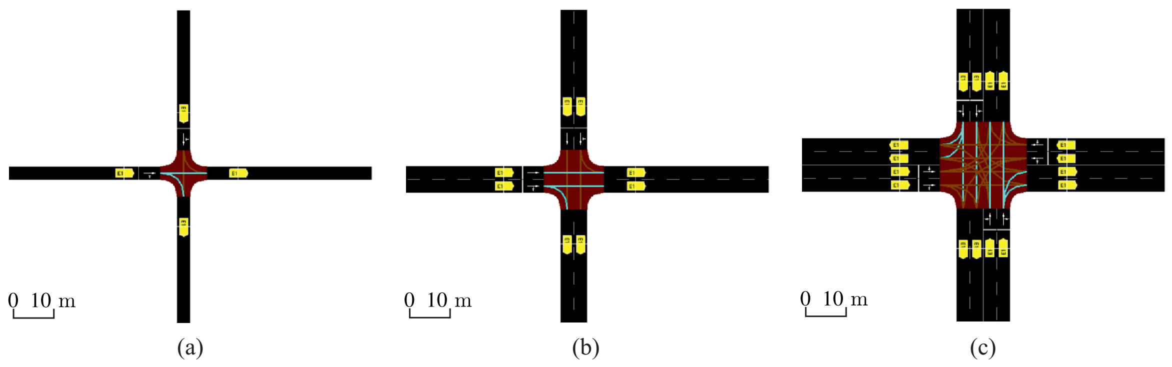

Figure 2. Networks of simulated junctions. Induction loops detectors are highlighted in yellow

Table 2. Groups of experiments for intersections a) and b) according to the relations of incoming/outgoing flows and splits

Experiment	Eq. inputs	Eq. splits	Eq. outputs
1	+	+	+
2, 3	+	–	+
4	–	+	+
5, 6, 7, 8	–	–	–

For each router we set the parameter of vehicle departure position at 0 and the departure position at “max”. For the `flowrouter` and the `routeSampler` we set the departure lane at “best”, while the `rlRouter` defines the departure lane by the corresponding agent. 1 minute data agregation was used for both `flowrouter` and `routeSampler`, since it is the smallest possible for these algorithms. The `rlRouter` was trained on data with 1 second aggregation. “–lane-based” additional parameter was used for the `flowrouter`.

Results

We used Mean Absoluter Error (MAE) as the measure of the deviation from both cumulative detectors and routes data.

The first junction (Table 3) was covered by only two starting lane-agents, but there are many interactions between agents, since the routes are less isolated from each other. In such a meshed network, the `routeSamler` provides much better routes compared to the `flowrouter`. However, without additional assumptions about routes distributions, the `routeSamler` uses uniform distribution for sampling routes. Therefore, it works better when routes splits are equal (experiments 1 and 4). At the same time, the `rlRouter` learns information about route splits by interacting with the network, which allows it to outperform the `routeSamler` in all experiments.

The second junction (Table 4) is covered by four lane-agents, but their routes intersect less, compared to the first intersection, due the separate lanes. In this case, the `flowrouter` performs better and surpasses `routeSamler` in a number of experiments. However, the `rlRouter` still provides the best routes.

The third intersection (Table 5) contains eight lane-agents with a high level of entanglement. The `rlRouter` has the smallest error for all routes, while the `flowrouter` and the `routeSamler` provide unstable results.

In all experiments the `rlRouter` showed compliance with detectors an order of magnitude higher compared to the `flowrouter` and the `routeSamler`.

Table 3. MAE for the cumulative number of routes at the junction a)

Route	Router									
		p_{UD} =	0.075	0.05	0.1	0.1	0.025	0.05	0.0375	0.0375
		p_{UR} =	0.075	0.1	0.05	0.1	0.05	0.025	0.0375	0.0375
		p_{LD} =	0.075	0.1	0.05	0.05	0.075	0.075	0.1	0.05
		p_{LR} =	0.075	0.05	0.1	0.05	0.075	0.075	0.05	0.1
UD	`rlRouter`		2.06	2.22	3.51	2.94	5.93	4.76	3.82	3.58
	`routeSampler`		15.79	42.26	28.14	3.25	14.14	13.01	11.12	13.01
	`flowrouter`		82.92	113.08	49.27	66.87	67.77	50.25	62.09	50.25
UR	`rlRouter`		2.02	1.71	3.59	2.83	5.38	4.58	4.01	3.87
	`routeSampler`		3.68	27.07	39.68	7.52	2.29	29.90	4.06	29.90
	`flowrouter`		70.34	99.36	38.83	61.24	55.32	34.20	47.19	34.20
LD	`rlRouter`		1.93	1.61	3.75	3.50	5.87	4.11	2.99	3.47
	`routeSampler`		6.72	31.32	36.37	11.83	8.43	21.87	3.23	21.87
	`flowrouter`		76.46	105.91	43.26	58.65	62.38	37.54	52.01	37.54
LR	`rlRouter`		2.02	1.77	3.99	3.10	5.87	3.91	4.19	3.67
	`routeSampler`		13.13	37.75	29.20	5.65	13.27	19.73	5.50	19.73
	`flowrouter`		83.94	112.74	50.21	73.48	68.13	39.95	54.23	39.96
Detectors	`rlRouter`		0.65	0.68	0.88	1.16	0.75	0.82	0.97	0.94
	`routeSampler`		10.60	11.00	9.20	10.38	9.3	9.53	8.96	9.53
	`flowrouter`		9.68	10.03	8.49	10.01	8.67	8.84	8.21	8.84

Table 4. MAE for the cumulative number of routes at the junction b)

Route	Router									
		p_{UD} =	0.15	0.1	0.2	0.2	0.05	0.1	0.075	0.075
		p_{UR} =	0.15	0.2	0.1	0.2	0.1	0.05	0.075	0.075
		p_{LD} =	0.15	0.2	0.1	0.1	0.15	0.15	0.2	0.1
		p_{LR} =	0.15	0.1	0.2	0.1	0.15	0.15	0.1	0.2
UD	`rlRouter`		8.35	11.95	2.39	7.89	10.08	3.33	7.39	1.41
	`routeSampler`		8.57	82.03	39.29	8.02	38.77	14.19	30.72	5.23
	`flowrouter`		28.00	54.73	10.13	30.46	27.25	26.36	58.43	3.79
UR	`rlRouter`		6.04	2.83	2.87	3.08	3.78	4.51	6.06	1.29
	`routeSampler`		18.05	49.64	64.52	20.4	7.63	41.81	3.07	36.46
	`flowrouter`		22.2	20.67	31.92	22.47	22.74	22.06	17.79	28.01
LD	`rlRouter`		3.33	3.06	1.25	8.19	3.29	1.60	8.91	1.26
	`routeSampler`		15.00	52.12	68.43	101.16	31.84	26.42	13.81	8.21
	`flowrouter`		15.28	20.88	19.98	24.58	8.56	15.21	11.70	12.92
LR	`rlRouter`		5.18	15.71	2.05	7.44	5.08	1.84	7.19	3.31
	`routeSampler`		10.47	84.06	40.13	78.60	29.26	22.67	15.41	7.33
	`flowrouter`		7.35	17.68	7.94	5.81	5.14	4.37	1.46	3.67
Detectors	`rlRouter`		2.13	1.21	1.82	2.92	1.57	1.61	3.74	0.78
	`routeSampler`		26.60	36.04	32.80	70.07	23.79	19.08	28.54	23.20
	`flowrouter`		14.63	13.11	13.72	14.56	11.93	14.43	15.13	13.63

Table 5. MAE for the cumulative number of routes at the junction c)

N	Router	Route	UL	UD	UR	LD	LR	LU	DR	DU	DL	RU	RL	RD
1	`rlRouter`		2.2	7.1	3.6	10.2	4.2	12.4	4.2	3.8	12.9	7.1	4.5	2.8
	`routeSampler`		8.0	55.6	44.6	16.2	55.8	56.2	5.9	51.4	42.1	15.3	48.9	51.9
	`flowrouter`		48.2	19.3	31.6	46.5	9.9	31.2	52.9	4.1	38.7	47.3	6.0	41.4

Note: for each starting edge, the probabilities of going right, straight and left are 0.01, 0.075 and 0.05, respectively.

Conclusion

The study presents a new method for generating routes in SUMO based on data from detectors. Unlike the existing algorithms, it uses multi-agent reinforcement learning and is trained directly in SUMO simulation, which allows it to use additional information about interactions within the road network. This approach made it possible to achieve a significant improvement in the observed metric (compliance with data from detectors) and obtain greater accuracy of the restored routes.

Unfortunately, the data from loop detectors impose a number of restrictions on the possibility of restoring the original routes along them. Using a simulation of the road network to extract additional information only partially overcomes these limitations. The extensive introduction of more informative transport sensors (camera detectors) is promising for further research in the field of intelligent transport systems.

References

Baumgart U., Burger M. Optimal control of traffic flow based on reinforcement learning // International Conference on Vehicle Technology and Intelligent Transport Systems. — 2021. — P. 313–329.

Behrisch M., Erdmann J. Route estimation based on network flow maximization // EPiC Series in Engineering. Conference: SUMO 2018 – Simulating Autonomous and Intermodal Transport Systems. — 2018. — Vol. 2. — P. 73–182.

Chouiekh Ch., Yahyaouy A., Aarab A., Sabri A. Road traffic: deep Q-learning agent control traffic lights in the intersection // 2022 International Conference on Intelligent Systems and Computer Vision (ISCV). — 2022. — P. 1–5.

Kheterpal N., Parvate K., Wu C., Kreidieh A., Vinitsky E., Bayen A. Flow: Deep reinforcement learning for control in sumo // EPiC Series in Engineering. Conference: SUMO 2018 – Simulating Autonomous and Intermodal Transport Systems. — 2018. — Vol. 2. — P. 134–151.

Liu Sh., Wang Yu., Chen X., Fu Yo., Di X. SMART-eFlo: An integrated SUMO-Gym framework for multi-agent reinforcement learning in electric fleet management problem // 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC). — 2022. — P. 3026–3031.

Lopez P. A., Behrisch M., Bieker-Walz L., Erdmann J., Flötteröd Y.-P., Hilbrich R., Lücken L., Rummel J., Wagner P., Wießner E. Microscopic traffic simulation using sumo // 2018 21st international conference on intelligent transportation systems (ITSC). — 2018. — P. 2575–2582.

MAVEN. — [Electronic resource]. — https://cordis.europa.eu/project/id/690727 (accessed: 25.12.2023).

Santana S. R., Sanchez-Medina J. J., Rubio-Royo E. How to simulate traffic with SUMO // Computer Aided Systems Theory – EUROCAST 2015: 15th International Conference, Las Palmas de Gran Canaria, Spain, February 8–13, 2015. — 2015. — Revised Selected Papers 15. — P. 773–778.

Seliverstov S., Seliverstov Ya., Gavkalyk B., Fahmi Sh. Development of transport infrastructure organization model for modern cities with growing effectiveness // Transportation Research Procedia. — 2020a. — Vol. 50. — P. 614–625.

Seliverstov S. A., Seliverstov Ya. A., Shatalova N. V., Korolev O. A., Borodina O. V., Kiselev A. A. Model development and assessment of a complex intersection of a road network using modern software systems // 2020 XXIII International Conference on Soft Computing and Measurements (SCM). — 2020b. — P. 92–96.

Wesemeyer D., Trumpold J. Controlling a real-world intersection with connected vehicle information provided by CAMs (Cooperative Awareness Messages) // EPiC Series in Computing. SUMO User Conference. — 2019. — P. 206–212.

Wu C., Kreidieh A. R., Parvate K., Vinitsky E., Bayen A. M. Flow: A modular learning framework for mixed autonomy traffic // IEEE Transactions on Robotics. — 2021. — Vol. 38, No. 2. — P. 1270–1286.

Zaky M., Airulla D. G., Joelianto E., Sutarto H. Y. Urban traffic simulation using SUMO open source tools // Internetworking Indonesia Journal. — 2017. — Vol. 9, No. 1. — P. 83–88.