

Traffic Simulation Environment Based on Sumo Software

Xuhong Li^{1*}, Liyong Zheng^{1†}, Bin Su¹,
Xu Guo¹, Yonggang Hao¹ and Wenjing Li¹

¹HANGZHOU HIKVISION DIGITAL TECHNOLOGY CO.LTD.

lixuhong@hikvision.com, zhengliyong@hikvision.com, subin5@hikvision.com,
guoxu5@hikvision.com, haoyonggang@hikvision.com, liwenjing8@hikvision.com

Abstract

In recent years, traffic modeling and simulation techniques have been widely used in the research on urban traffic planning, signal control optimization, and many other traffic-related subjects. Meanwhile, the extensive employment of traffic simulation in industry gave rise to numerous traffic virtual environments. In this paper, we introduced an easy-to-use simulation environment based on SUMO (SumoEnv). The SumoEnv can generate a wealth of traffic indicators and provide a simpler access to them by encapsulating the original SUMO application programming interfaces (API) into a more integrated and intuitive one. With this convenience, algorithm engineers who are not familiar with traffic simulation may save their precious time and effort while working with SUMO. Moreover, the SumoEnv is also equipped with a lot of traffic simulation scenario models to reduce the difficulty of using SUMO.

1 Introduction

With the rise of digital twin concepts and technologies, urban traffic simulation environments are becoming more and more popular in the industrial world. Up to now, traffic simulation is widely used in the traffic management and control, such as signal control, vehicle guidance, road channelization design, traffic incident simulation, etc. ^[1, 2, 3]. Typically, most simulation applications must rely on mature traffic simulation software, SUMO or VISSIM for example. ^[4] As a result, when developing algorithms in the environments of this software, it is necessary to manually build credible traffic simulation models. On the other hand, corresponding programs are needed to monitor the interactions between the algorithm and the simulation environment. Unfortunately, both of these two works are very time-consuming and labor-intensive, with the outcomes of them hardly to be reused effectively.

In current situation, there is a strong demand for traffic simulation. However, the construction of simulation models and the development of interactive environments raise a high barrier between the professionals and others, which is time-consuming and labor-intensive to overcome. This unbalanced relationship reflects the status quo of traffic simulation applications and seriously restricts the rapid practice of them. Therefore, we are committed to develop a complete and universal traffic simulation environment to solve this problem. The traffic simulation environment consists of an interactive environment and a scenario library. Many traffic indicators are ready to be retrieved from the interactive environment through only a few interfaces. In addition, the simulation scenario library covers several standard simulation models such as single intersection, trunk and region model, as well as the scenarios gathered in practical application process.

More recently, the deployment of reinforcement learning (RL) in the development of traffic management and control algorithms further spawned some associated traffic virtual environments, such as the CityFlow maintained by Shanghai Jiaotong University ^[5], and the flow project maintained by Berkeley University ^[6]. These projects construct a RL framework in the simulation environment, providing a large-scale city traffic scenario for the training of traffic optimization algorithms. However, these environments are strongly tied with RL itself, which means that there would be a large number of complex configurations and strict constraints. One must handle a large amount of complicated coding work first before creating a single traffic scenario, and has no access to obtain additional traffic indicators. This makes the virtual environments very unfriendly to users who have little experience in using traffic simulation. Worse still, the communities of these projects are not very active, leaving the out-of-date versions and a lack of documentation.

To fill the abovementioned gap, we are aiming to develop a generic and simple simulation environment that can provide a wealth of traffic indicators and support the expansion of custom ones. Certainly, fewer and simpler interface functions are preferred to facilitate the interaction between users and the simulation environment. In addition, the environment is expected to have the capabilities of traffic incident simulation. Along with a scenario library that contains many traffic models, this simulation environment may significantly reduce the cost of time and effort for users when using traffic simulation.

Therefore, with the goal of providing better services for practical applications, we built a simulation environment based on the secondary development of SUMO. Furthermore, we are planning to contribute this work to the SUMO community. The main characteristics of it are as follows:

- 1) An out-of-the-box virtual traffic environment, which is able to provide a wealth of traffic indicators according to the configuration of the user;
- 2) A few easy-to-use APIs to interact with the traffic simulation environment;
- 3) A traffic simulation scenario library including many pre-defined simulation models;
- 4) Compatibility with SUMO's TraCI and Libsumo interfaces;
- 5) A web-based visualization interface for simulation indicators;
- 6) Ability to simulate traffic incidents.

2 Design Concepts

Basically, there are two interactive interfaces named TraCI and Libsumo in the SUMO software. They already present many functions for users to obtain several traffic indicators or interact with the traffic lights and vehicles in simulation environment. Despite this, to fully utilize the TraCI or Libsumo, users have to be familiar enough with not only SUMO, but also some common sense of traffic simulation, which raises a high barrier for normal algorithm engineers.

In order to reduce the difficulty for users to use SUMO, we re-encapsulate the SUMO software into a simulation environment (SumoEnv) with the concept of object-oriented programming (OOP).

Although the SumoEnv exposes only a few simple interface functions, it is able to provide users with access to obtain all traffic indicators, control traffic lights, or simulate traffic incidents. Additionally, it is equipped with a built-in simulation scenario library, so users may have no need to build sumo simulation models by themselves.

The traffic indicators in the SumoEnv are designed as attributes attached to the lanes. Users can decide which traffic indicators will be computed and how to compute them by setting parameters in the SumoEnv's configuration (JSON files). Users can choose the calculation logic of indicators freely, and the SumoEnv will load corresponding indicator operator according to the strategy mode. Considering the efficiency of the algorithm development, a coroutine is deployed to allow transferring program control between the SumoEnv and the investigated algorithm. The design outline is shown below:

- a) An initialization function that integrates features such as parsing the road network file, initializing the lane class, and loading the configuration file, is presented.
- b) We adopted the idea of strategy combination in the calculation of traffic indicators to make sure that the SumoEnv loads only the required operators. By inheriting from the superclass, users are allowed to customize their own indicator calculation operators.
- c) The calculation of traffic indicators is designed as a method of the lane class. Parallel computing is employed here to reduce the calculation time.
- d) The simulation is conducted with coroutine that allows transferring program control between the SumoEnv and the investigated algorithm.
- e) An interface is reserved for interacting with the traffic lights. And it supports the signal control schemes with both the dual ring and signal group protocol.
- f) A function is designed in the SumoEnv for easy and dynamic generation of traffic incidents during the simulation. Most common traffic incident scenarios such as vehicle accidents, lane closures, and speed limits are covered by it.
- g) A web-based visualization interface is provided to display traffic indicators and quickly evaluate the simulation results.

Both the original TraCI and Libsumo interfaces are supported by the SumoEnv, which means that the SumoEnv can meet the requirements of both simulation animation effects and speed. Moreover, the adoption of the strategy mode not only guarantees the principle of subscribing traffic indicators on demand, but also saves computing resources and improves simulation efficiency. Meanwhile, plenty room is reserved for the expansion of the indicator calculation, which facilitates the subsequent maintenance of SumoEnv.

3 Brief Description

The SumoEnv is designed with the concept of OOP and implemented by python based on the open-source SUMO. The virtual environment is available to users via source code or python wheel package. In this section, the framework of the environment, classes unified modeling language (UML), API presented to users, and the interaction logic between the environment and external algorithms are described.

3.1 System Framework

SUMO is a microscopic traffic simulation software that can handle large networks and multi-modal random traffic. It supports intermodal simulation including pedestrians and comes with a large set of tools for scenario creation. It was mainly developed by employees of the institute of transportation systems at the German Aerospace Center. In fact, simulation software is an ideal tool for algorithms research, except for the difficulty for users who are not familiar with it to get started. Based on the

SUMO simulator, the SumoEnv was developed to provide an easy-to-use simulation environment for algorithm engineers. The four-layer framework of the SumoEnv is shown in Figure 1.

The first layer of the framework is a simulation scenario library that contains many standard SUMO network files and models calibrated with real-world data. Scenarios like single intersection, trunk, and region are all covered. The second layer is the SUMO simulator engine. As for the third layer, a simulation environment including road network parsing, signal control interaction, traffic indicators computing, and traffic incidents simulation functions is presented. These functions are easily accessible through only one simple API that is re-encapsulated from the original TraCI and Libsumo. Therefore, in the last layer, algorithm engineers can develop, train, and validate their algorithms without any concern about the simulation software. Additionally, the left side of the framework is the overall configuration of the virtual environment, while the right side represents the visualization for traffic indicators.

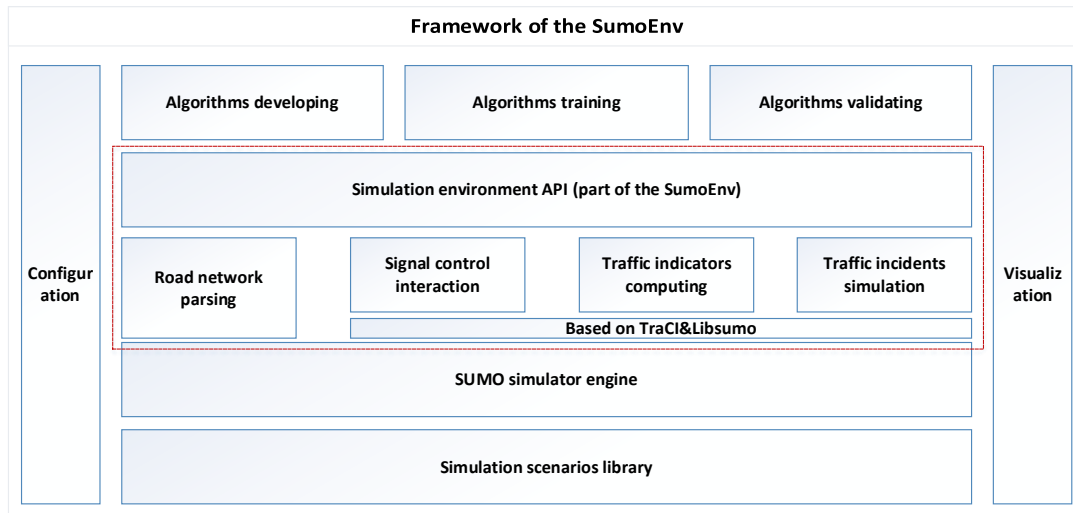


Figure 1. Framework of the SumoEnv

3.2 Object Structure of the SumoEnv

There are six main classes in our simulation environment, namely the SimEnv class, NetInfo class, LaneStruct class, TrafficLightHelper class, TrafficIncidentHelper class, and IndicatorAction class respectively.

All public functions available to users are included in the SimEnv class, the initialization and interaction functions for example. The NetInfo class undertakes the work of detecting and parsing the simulation network and additional files (*.net.xml and *.add.xml files in the SUMO model). Then, lane static information are extracted to create the LaneStruct class instances. The TrafficLightHelper provides the approach to control the signal lights, while the TrafficIncidentHelper enables the simulation of traffic incidents, such as lane closure, vehicle emergency stop, and lane speed limit. Finally, all traffic indicators provided by the SumoEnv are accessible in the IndicatorAction class that defines the logic of traffic indicators computing. The UML of those six classes is shown in Figure 2.

Since the NetInfo, TrafficLightHelper, and TrafficIncidentHelper class are just a plain combination of the raw SUMO APIs, only the SimEnv class, LaneStruct class, and IndicatorAction class will be described in detail here.

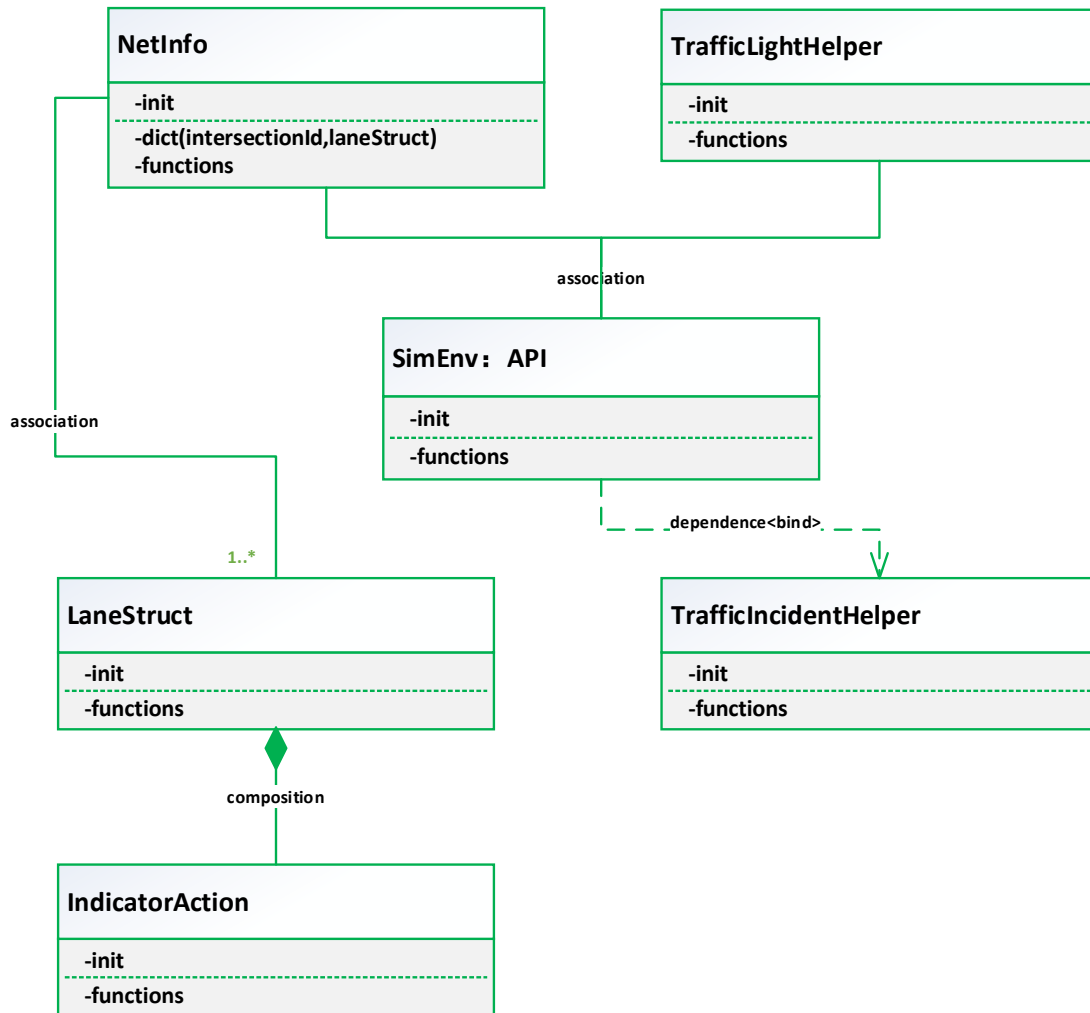


Figure 2. UML Structure

3.2.1 The SimEnv Class

The functions of the SimEnv class may be classified into four categories: basic simulation control, signal control, traffic incidents simulation and indicators computation. They are introduced in order as follows:

1) Basic simulation control

- Configuration

The configuration is formatted as a JSON file that declares some options about the SUMO simulator, detector types, indicators subscribed, and the computing logic of the indicators. An example of the configuration file is shown below.

```

{
  "auto_quit": true,
  "no_warning": true,
  "no_step_log": true,
  "num_threads": 1,
  "random": false,
  "route_steps": 200,
  "scale": 1.0,
  "e1Det":{"19":"LAST_STEP_OCCUPANCY",
           "23":"LAST_STEP_VEHICLE_DATA"},
  "e2Det":{"16":"LAST_STEP_VEHICLE_NUMBER",
           "20":"LAST_STEP_VEHICLE_HALTING_NUMBER",
           "18":"LAST_STEP_VEHICLE_ID_LIST"},
  "laneDet": {"122":"VAR_WAITING_TIME",
              "90":"VAR_CURRENT_TRAVELTIME",
              "51":"LANE_LINKS"},
  "Indicators":["PassVolume","OriginLaneQueue",
               "LaneVehicleNum","StopVehRate",
               "LoopOccupancy","TravelTime",
               "TravelSpeed"]
}

```

- Environment initialization

While initializing, users first have to choose which API (TraCI or Libsumo) to use. After the SUMO model file (*.sumocfg file) is loaded, the NetInfo class and LaneStruct class will be initialized. Then, some functions of those two classes will be called during the initialization of the SimEnv class to create a simulation environment instance.

- Simulation step

The simulation step of the SumoEnv is a wrapper of the SUMO's step function. During a simulation step, many tasks would be executed, including calling the current_step_evaluation() function of the LaneStruct instance to compute all the traffic indicators specified in the configuration.

2) Signal control

- Signal control schemes extraction

A function is offered for users to extract signal control schemes from the simulation environment and save them in the format of the LanePhaseStruct class. Since not all signal plans in SUMO can be converted to the dual ring structure successfully, the saved schemes are not presented with that structure. The structure of the LanePhaseStruct is shown as below.

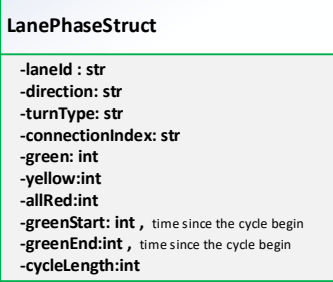


Figure 3. Structure of the LanePhaseStruct

- Interaction with traffic lights

As shown in Figure 4, the SumoEnv supports two kinds of signal control scheme protocols: the dual ring and signal group structure. Users can set up a new signal control scheme in the SUMO simulator via API of the SumoEnv.

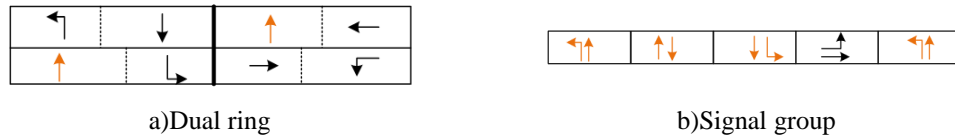


Figure 4. Two kinds of the protocols

Figure 4 shows the same signal control plan defined in different structures, i.e., dual ring and signal group. Nevertheless, as shown in Figure 5, some special signal control schemes can only be represented by the signal group structure.



Figure 5. An example of the scheme

As signal control schemes in real world are represented by either dual ring or signal group structure, the SumoEnv is thus designed to be compatible with both of the protocols. Once the scheme data is loaded in the SumoEnv, a conversion process would be conducted to obtain control data with the logic structure of the SUMO. Thereafter, the traffic lights' states may be changed according to it. The detailed logic of the conversion is defined in the TrafficLightHelper class.

- 3) Traffic incidents simulation

The TrafficIncidentHelper enables the simulation of traffic incidents, such as lane closure, vehicle emergency stop, and lane speed limit. These incidents simulation functions can also be supported by the original SUMO APIs.

- 4) Indicators computation

Traffic indicators are designed as the attributes attached to the LaneStruct class. They are computed every simulation step and then aggregated over the time interval specified by users.

- Current-time indicator

Part of the indicators named current-time indicators will be computed every simulation step, which is triggered simultaneously by the simulation step function. That is to say, when we call the simulation step function, the attributes of all the LaneStruct instances in the SumoEnv will update. Moreover, different computing logic are supported in the IndicatorAction class to generate different indicators. Detailed information about this can be found in the "The IndicatorAction Class" section.

- Time-interval indicator

The SumoEnv has the flexibility to adjust how long the time interval for indicators aggregation is, depending on the input of the SumoEnv.get_interval_data function. In order to reduce the memory and time consumption of the computation, an incremental computing method is adopted. Again, the detail will be described in the following section.

3.2.2 The LaneStruct Class

According to their variability, the attributes of the LaneStruct class can be divided into two categories: static and dynamic information of the lane. The lane's static information includes ID, length, direction etc., whereas the dynamic information includes the current-time and time-interval indicators: volume, queue length, delay, stops, travel time, and travel speed for example. The computing logic of those indicators is shown in Figure 6.

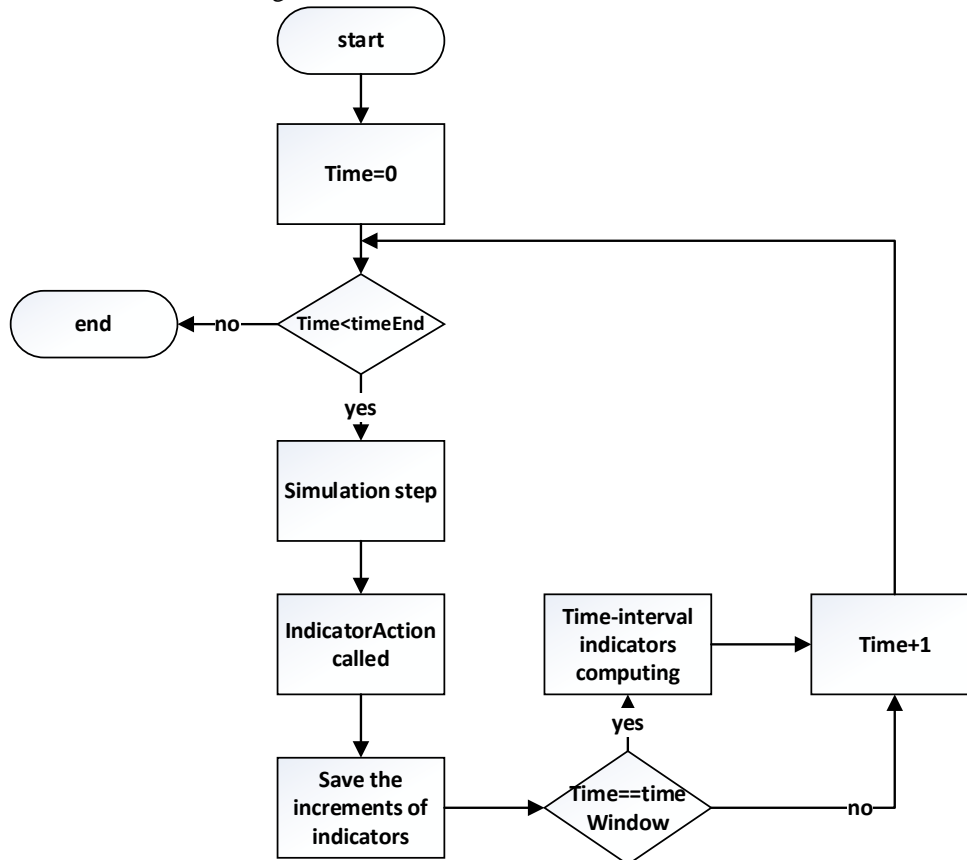


Figure 6. Computing logic of the indicators

Rather than all the history values, only part of the changed indicators will be saved by the incremental computing method. Taking the calculation of volume as an example, at the beginning of each time interval, the volume variable is cleared and only the cumulative volume from then on will be recorded.

3.2.3 The IndicatorAction Class

The IndicatorAction is a superclass designed for indicator computing. As shown in Figure 7, the computation of indicators like the pass volume and queue length is achieved by inheriting the superclass to rewrite the computing logic of the indicator. The SumoEnv provides some pre-defined subclass of the IndicatorAction superclass. Meanwhile, users can also implement a new subclass to overwrite a

customized computing logic of the indicator. More considerably, the IndicatorAction is attached to the LaneStruct class in a way of combination to make sure that users can choose the computing logic freely. As for which subclasses of the IndicatorAction would be loaded onto the LaneStruct class, it depends on the configuration file of the SumoEnv.

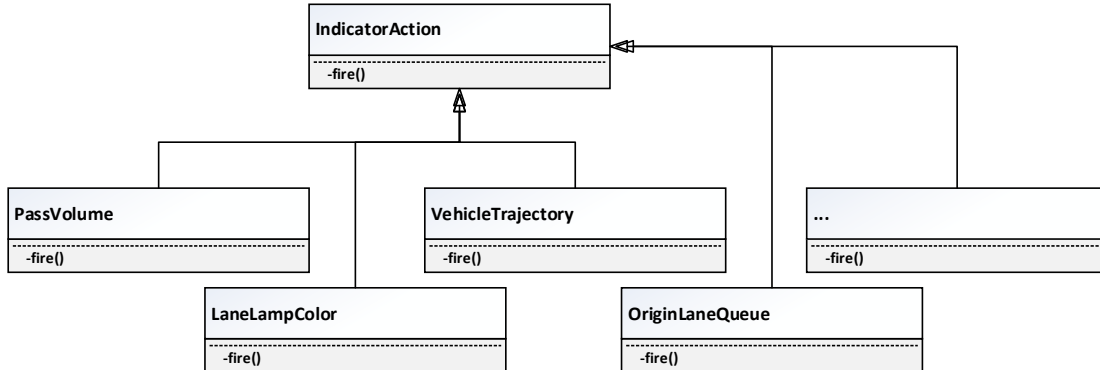


Figure 7. The relationship of indicators' algorithms

3.2.4 Interfaces of the SumoEnv

Tables 1 & 2 show all the pre-defined traffic indicators and functions provided for users respectively. By default, traffic indicators are computed in lane dimension. Still, users are allowed to compute them in other spatial dimensions, for example in the road, intersection, or network dimension. Furthermore, the latter visualization module is able to display all of these indicators.

| index | function signature | parameters | description |
|-------|--------------------------|---|--|
| 1 | get_interval_data(tlsId) | tlsId->string | get statistics for all indicators within the time interval window |
| 2 | veh_stop() | vehId->string laneId->string position->float duration->int | make a specified vehicle stop at the given position in the lane for a given duration of time |
| 3 | lane_close() | laneId->string | close a specified lane |
| 4 | lane_resume() | laneId->string | reopen a lane if it is closed |
| 5 | lane_speed_limit() | targetRoad->string targetSpeed->double isEdge->bool (default: true) | limit the speed of a specified road or lane to the targetSpeed (m/s) |
| 6 | load_dual_ring() | tlsId->string pattern->DualRingStruct object | load a dual-ring signal control plan into the SUMO |
| 7 | load_signal_group() | tlsId->string pattern->FlowSchemeStruct object | load a signal group (traffic flow) control plan into the SUMO |
| 8 | switch_phase() | tlsId->string phaseLst->[PhaseStruct,...] | switch the phase of a specified set of traffic lights according to the PhaseStruct |
| 9 | run_step(parallel=True) | None | run a simulation step |

Table 1: Functions for users provided by the SumoEnv

| index | indicator name | description |
|---|-----------------------------|---|
| 1 | current_arrival_volume | the number of vehicles that arrived at the upstream of the lane within the last simulation step |
| 2 | current_pass_volume | the number of vehicles that passed the stop line of the intersection within the last simulation step |
| 3 | current_lane_veh_num | the number of vehicles that were on the lane within the last simulation step |
| 4 | current_queue_num | the number of queued vehicles that were on the lane within the last simulation step |
| 5 | current_queue_length | the sum of queued vehicles length that were on the lane within the last simulation step |
| 6 | current_waiting_time | the mean waiting time of all vehicles that were on the lane within the last simulation step |
| 7 | current_queue_ratio_of_lane | the proportion of queue length to lane length within the last simulation step |
| 8 | current_travel_time | the mean travel time of vehicles that were on the lane within the last simulation step |
| 9 | current_travel_speed | the mean travel time of vehicles that were on the lane within the last simulation step |
| 10 | current_stops | the stop time of vehicles that were on the lane within the last simulation step |
| 11 | current_nonstop_rate | the proportion of non-stop vehicles to total vehicles that were on the lane within the last simulation step |
| 12 | current_lamp_color | the lamp color(green/yellow/red) of the lane controlled by traffic light within the last simulation step |
| 13 | current_veh_trajectory | the position (x, y, z) of vehicles that were on the lane within the last simulation step |
| 14 | current_lane_density | the vehicle density on the lane within the last simulation step |
| statistics of the time-interval indicators | | |
| 15 | interval_arrival_volume | the total number of vehicles that arrived at the upstream of the lane in the interval |
| 16 | interval_pass_volume | the total number of vehicles that passed the stop line of the intersection in the interval |
| 17 | interval_max_queue_num | the max number of queued vehicles in the interval |
| 18 | interval_max_queue_length | the max queue length in the interval |
| 19 | interval_mean_queue_num | the mean number of queued vehicles in the interval |
| 20 | interval_mean_queue_length | the mean queue length in the interval |
| 21 | interval_wait_time | the max waiting time of all vehicles in the interval |
| 22 | interval_mean_travel_time | the mean travel time in the interval(second) |
| 23 | interval_mean_travel_speed | the mean travel speed in the interval(m/s) |
| 24 | interval_mean_stops | the mean number of vehicles stopped in the interval |
| 25 | interval_mean_nonstop_rate | the mean 'current_nonstop_rate' in the interval |

Table 2: Pre-defined indicators for users provided by the SumoEnv

3.2.5 Interaction Logic

As shown in Figure 8, users can interact with the SumoEnv via python APIs in the SimEnv class listed in Table 1.

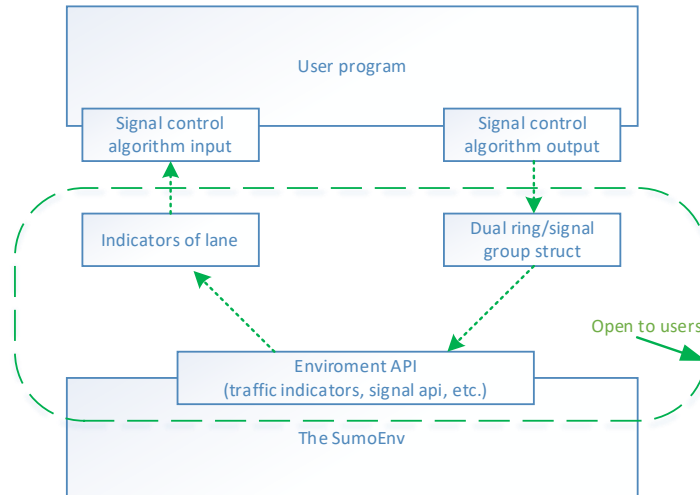


Figure 8. logic of interaction

Take the development of the signal control optimization algorithm as an example. During the simulation, algorithm engineers can easily access the traffic indicators they need with the `get_interval_data` function. These indicators are fed back into the optimization algorithm to iteratively generate an adjusted signal control plan. Then, this new plan will be loaded into the SumoEnv to interact with the simulation environment and yield new indicators. More specifically, the inputs of the optimization algorithm are queue length and pass volume, and the outputs are signal control plans in the form of dual ring or signal group protocol. On the contrary, the signal control plans are inputs of the SumoEnv and the traffic indicators are outputs. In the optimization process, traffic indicators and signal control plans serve as a bridge between the simulation environment and the algorithm. The overall process is shown in Figure 9.

Prepare in advance: choose a SUMO model from the simulation scenario library or build a new model

step0: initialize the SumoEnv (load SUMO model and configuration)

step1: while sim_time < endTime

step1.1: simulation step

step1.2: call the get_interval_data function to compute time-interval indicators if a time interval ends, else do nothing

step1.3: call signal control optimization algorithm to generate a new signal control plan if the algorithm is triggered

step1.4: call the load_dual_ring function to load the signal control plan into the SUMO if the algorithm returned a new one, then loop step 1

step2: end loop, then visualize the traffic indicators

step3: end simulation

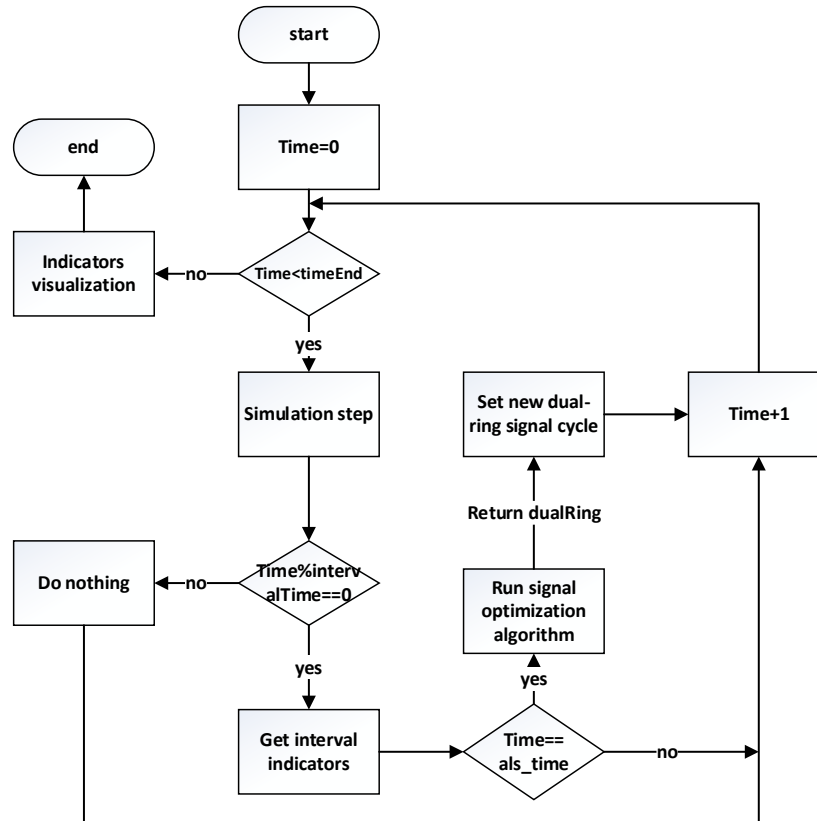


Figure 9. process of algorithm

4 Application

The SumoEnv can provide a virtual environment for traffic algorithm developing, training and testing in many different simulation scenarios. Both the development of the traditional and reinforcement-learning-based signal control algorithms may be conducted in the SumoEnv. Moreover, other work, such as observing the evolution of the traffic flow under incidents and evaluating the optimization of the road channelization, is also available on it.

4.1 Algorithm Development and Validation

4.1.1 Isolated-intersection and Arterial-coordinated Signal Control Algorithm Development

The evaluation of the development of isolated-intersection and arterial-coordinated signal control algorithm is shown in Figure 10. On the left, red circles and boxes highlight the targeted intersections

and arterials in the road network. And the charts on the right show the improvement in each indicator, including the maximum queue length, average stops, time delay, and travel speed.

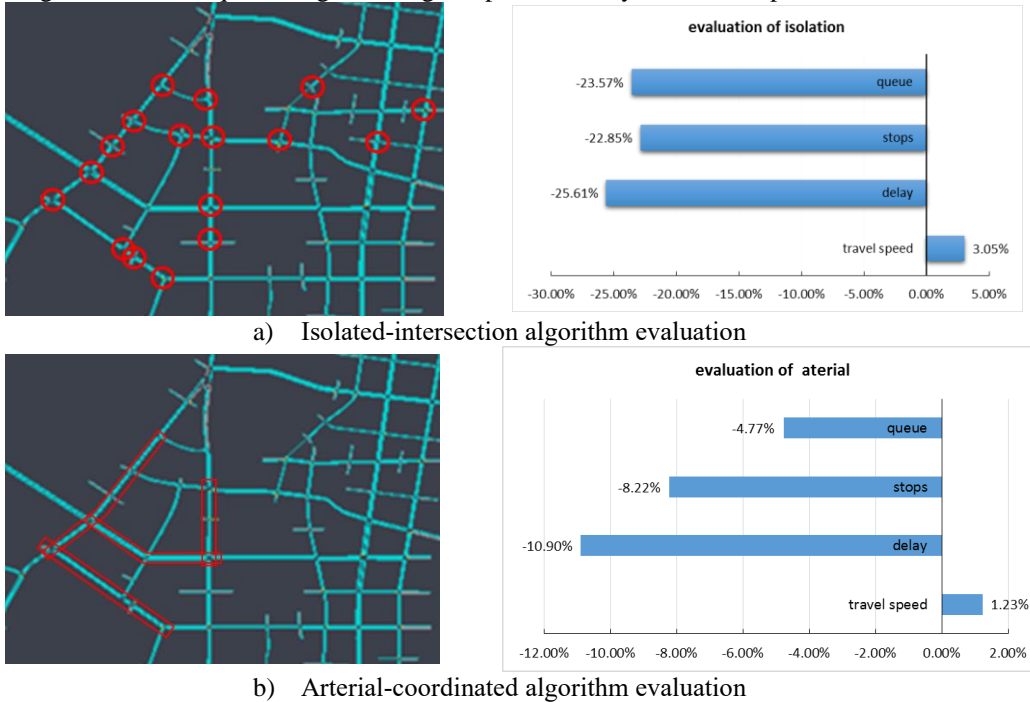
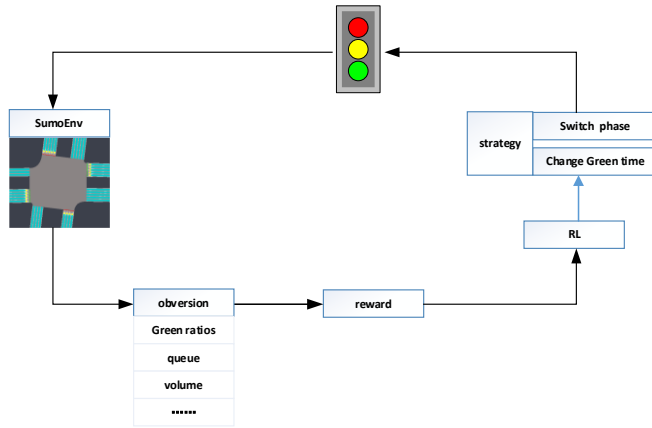


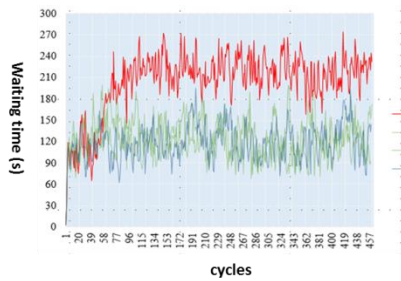
Figure 10. Signal control algorithm development and validation in the SumoEnv

4.1.2 Reinforcement-learning-based Signal Control Algorithm Development

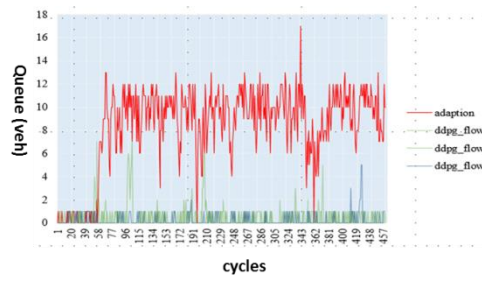
The employment of the reinforcement learning method in the development of signal control algorithm makes the algorithm more intelligent. In general, the RL works by obtaining traffic indicators from the SumoEnv and then generating an adjusted signal control plan to interact with the simulation environment, which is shown in Figure 11 a). RL algorithm obtains the traffic state from SumoEnv, calculates a new signal control scheme, and sends it to SumoEnv. As the simulation results we can see in Figure 11 b) and c), the RL-based algorithm is superior to the traditional adaptive algorithm.



a) Framework of the reinforcement learning method



b) Average waiting time of vehicles

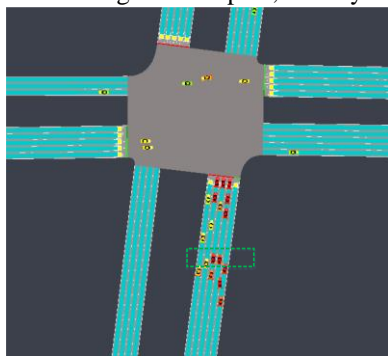


c) Queue length

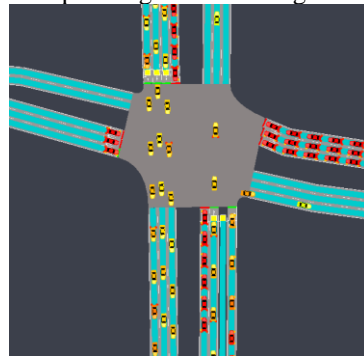
Figure 11. Reinforcement learning in simulation

4.1.3 Traffic Incidents Simulation

As shown in Figure 12, simulation of incidents like vehicle emergency stop and lane closure may be carried out in the virtual environment by functions of the SumoEnv's TrafficIncidentHelper class. Furthermore, by predicting the evolution of the traffic flow after incident, the signal control plan could be adjusted to mitigate its impact, namely the incident-based adaptive signal control algorithm.



a) Car-stop-incident



b) Lane-closure-incident

Figure 12. Traffic incidents simulation in the SumoEnv

4.2 Visualization of the Traffic Indicators

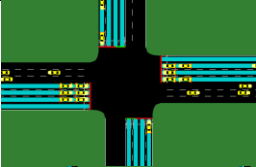

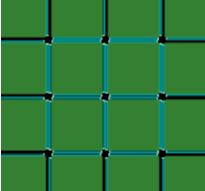
In addition, a web-based visualization module in the SumoEnv is designed to realize the visualization of the traffic indicators. With the traffic indicators obtained from the SumoEnv and saved as csv files, the visualization module will read and show them in charts with different spatial dimensions on the web page, as shown in Figure 13.

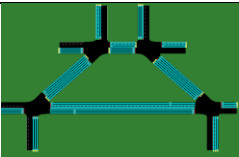
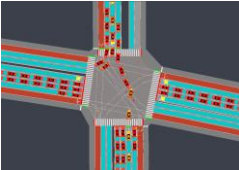
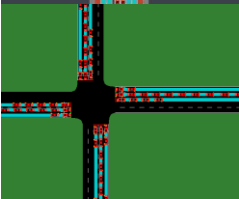
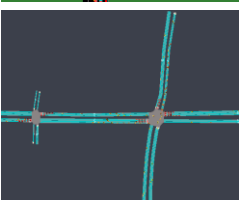
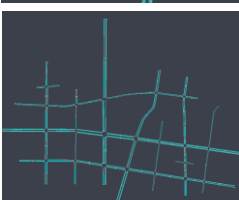
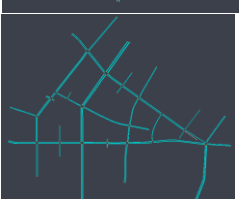
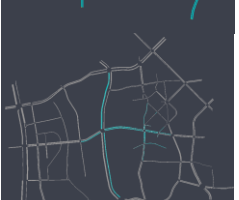


Figure 13. Visualization in different spatial dimensions

4.3 The Simulation Scenario Library

Finally yet importantly, the SumoEnv is equipped with a simulation scenario library that covers standard traffic models built with SUMO and models calibrated with observed data in real world. More calibrated models will be continuously uploaded in the future.

| index | name | screen |
|-------|---|--|
| 1 | isolated-intersection-with-double-hump-flow |  |
| 2 | long-short-arterial |  |
| 3 | normal-network |  |

| | | |
|------------------------------|--|---|
| 4 | abnormal-network |  |
| models with real data | | |
| 5 | isolated-intersection -with-peds-and-bikes -with-real-data |  |
| 6 | isolated-intersection -with-real-data |  |
| 7 | two-intersection -with-real-data |  |
| 8 | middle-network -with-real-data |  |
| 9 | middle-network -calibrated-with-real-data |  |
| 10 | complex-network |  |

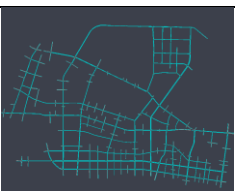
| | | |
|---|---|--|
| 11 city-network -with-real-data |  | |
|---|---|--|

Table 3: Simulation scenarios in the library

Note: Induction loops (E1) and lane-area detectors (E2) are deployed by default in all models.

5 Conclusion

In this paper, we introduced an easy-to-use SUMO-based simulation environment called the SumoEnv. By presenting its development objective, design concepts, logical framework, traffic indicators computation, and specific application cases, we are trying to help you understand it and get started quickly. A step further, the fewer and simpler APIs provided by it are beneficial to algorithm engineers who are not familiar with the SUMO and traffic simulation. Besides, the SumoEnv is equipped with a simulation scenario library, which minimizes the time-consuming work of building simulation models. Moreover, the web-based visualization module facilitates the graphical presentation of traffic indicators in multiple spatial dimensions. Since the SumoEnv has provided a set of simple API to interact with the simulation environment, a large number of built-in scenarios ready to be used, and a visualization module to display traffic indicators intuitively, algorithm engineers may entirely concentrate on their algorithm without any concern.

In summary, the SumoEnv is an out-of-the-box traffic simulation environment. Currently, it is compatible with the SUMO version 1.2-1.8 and python 3.6-3.7. Moving forward, we plan to contribute it together with the simulation scenario library to the community in the form of source code and python third-party package.

References

- [1] Mahajan S K, Umadekar A, Jethwa K. New Concept of Traffic Rotary Design at Road Intersections [J]. Procedia - Social and Behavioral Sciences, 2013, 96:2791-2799.
- [2] Pei J. Application of Vissim in the Traffic Simulation of Changzhou Integrated Passenger Hub. Journal of Transport Information and Safety, 2010.
- [3] <https://sumo.dlr.de/docs/Other/Projects.html>
- [4] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 2575-2582, doi: 10.1109/ITSC.2018.8569938.
- [5] Zhang H, Feng S, Liu C, et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. 2019.
- [6] Wu C, Kreidieh A, Parvate K, et al. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. 2017.