Universiteit Utrecht

Generating Traffic for the Traffic Simulator

Zvezdelin Stoyanov
1094459

Master thesis
July 2022

Supervisor: Prof. Dr. I. Velegrakis
Second Supervisor: Ir. O.A. Vermeulen
Second examiner: Dr. I.R. Karnstedt-Hulpus

Abstract

The goal of our work was to create a traffic simulation environment where vehicles are moved across a road network. Cars can have different settings that change their decision making process.

The simulator requires accurate data in order to achieve meaningful results. This paper describes the traffic generation for our simulator. The script has different tools for generating car data, according to the required purpose of the simulation. The described work in this paper is capable of generating cars starting from specified areas ending in other specified areas, generating flows of cars, using real data to create realistic synthetic data, and generating data for simulating events.

Table of Contents

Introduction

Purpose

The simulator can be used in experiments for various purposes, such as traffic forecasting, evaluation of road capacity and analyzing traffic.

The Traffic Generator has a key role in successfully using the simulator. Adding cars manually is not possible for large-scale scenarios and a data generation script is required. This script allows for creating big lists of cars – their starting point, their ending point, and their departure time. Their arrival time is not a part of the generated data, as it is to be calculated by the simulator.

Using the simulator together with the Traffic Generator allows for optimization of traffic systems. It is faster and cheaper to use the simulator than performing real-world testing experiments.

Alternative Simulators

There exist alternative packages to our simulator. CityFlow and Simulation of Urban Mobility (SUMO) are two of the most widely used ones. The reason for the creation of CityFlow was that SUMO has slower performance for large road networks and large traffic flow. CityFlow has optimized data structures and more efficient algorithms than SUMO. CityFlow is more efficient but it is not more effective. The difference in effectiveness (how accurate the simulation is) between the two is minimal and SUMO remains the number one most popular package for simulating traffic. It has over 20 years of history and has been used in numerous international projects.

Comparison to "Simulation of Urban MObility" (SUMO)

SUMO’s documentation describes the package as “an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks”. The script we have created shares some features with SUMO. The common features between our traffic generator and what the creators of SUMO call their “demand modelling” are the following: both SUMO and our script can be used to create random traffic. Both can be used to create flows. Both can use real data – this happens by using “detector data” in SUMO, which is taking information from observation points and then using this data to generate demand. Using real data can be done with our script thanks to SDV – a package for creating realistic synthetic data. Both my generator and SUMO can generate cars from specific areas leading to other specific areas – this can be done in SUMO by using the OD matrices (origin-destination matrices), and a similar effect can be done in our script by using the edge probabilities.

An advantage of our script over SUMO is that our traffic generator can create traffic for simulating events – such as football matches, concerts, or other occasions where multiple cars end up in a single location in a specific time, or multiple cars leave from a single location in a specific time. In order for a user to achieve such an effect in SUMO, he would have to manually use the existing tools in order to do so. By using my generator, the user can use the two functions for events, which can handle the car generation for events automatically.

Another advantage of our script over SUMO is the option to choose which model to use when using real data to generate synthetic data. The user can choose from 4 different machine learning models. An evaluation framework is also given, so that the user can see which model gives the most accurate results.

Different Methods for Generating Traffic

1. Generating Cars Based on Edge Probabilities

1. Setting Edge Probabilities

The first data generating method works by generating data based on pre-defined probabilities. Every edge gets assigned an individual probability of being a start point or being an end point. We have created two starting settings – a homogenous distribution, where every edge gets assigned the same probability, and a heterogeneous distribution, where every edge gets assigned a random probability. After running one of the two functions, the script outputs two lists – the edges' probability of being a starting location and their probability of being an end location. Furthermore, there is function that can be used to adjust entries in the lists. This way the user can individually change the probabilities of edges of interest.

2. Generating Cars

After defining the probabilities of each edge, we can proceed with the actual generation of the data blocks or cars. The generator has two functions. It can generate cars/blocks according to starting probabilities or ending probabilities. The code works by generating a random value between 0 and 1 at each tick of the total time, and compares that random value with the previously assigned probability of the edge being a start/end point. If the probability of the edge is larger or equal to the generated random value, a car is created. If the probability is less than the random value, no car is created. In other words, if we set the probability of an edge being a starting point = 1, and we run the function generating cars according to starting probabilities, it will generate a car on that edge at each tick, as the probability is 100%.

2. Generating Cars Using Flows

An alternative way to generate cars/blocks is by using flows. A flow is defined as a car followed by other cars that appear in the same position at later points of time. Flows can be used together with the previous generator in order to add more cars, or they can be used separately, if the flows are sufficient for the simulation and the user does not require other traffic.

The first type of flow is a timed flow. By using it a car appears at each tick and follows the previous cars. The tick size can be adjusted, in order to regulate how often cars appear.

The second type of flow is a flow with probability. It works by generating a random value between 0 and 1 and comparing the random value to the defined probability. If the defined probability is larger than the random value a car is created. This comparison is done at each tick. If we set the probability to equal 1, that would create the same result as the timed flow, as that would create a car every tick. Tick size is also modifiable.

3. Generating Cars Using Real Data

There is a possibility for the user to run a simulation imitating real data. This can happen by importing a file containing the real data of cars (their starting position, their ending position and their starting time). After importing, the user can generate synthetic data by using the functions of SDV – the Synthetic Data Vault. This is a library that aims to generate new synthetic data that has the same format and statistical properties as the original dataset. The synthetic data is generated by using one of the following models:

- Gaussian Copula - It allows us to describe the distribution by analyzing the dependencies between their marginal distributions
- CTGAN - Based on the GAN-based Deep Learning data synthesizer
- TVAE - Triplet-based vibrational auto encoder
- CopulaGAN - variation of the CTGAN Model which takes advantage of the CDF based transformation that the GaussianCopulas apply to make the underlying CTGAN model task of learning the data easier

The user can choose which model to use, based on the accuracy of the synthetic data. The accuracy of the synthetic data is evaluated and returned by the function. The evaluation function works by applying different metrics and returns the average of the scores obtained in each one of them. The output is a number between 0 and 1, and the closer it is to 1, the more similar the synthetic data is to the real data.

4. Causing Events

1. Outgoing Event

An outgoing event is an event in which there is a single starting point of multiple cars. The difference between an outgoing event and a flow is that in a flow all cars have the same direction, however in an outgoing event the cars can have different directions. The user has to specify the location of the event - which is the starting point for the cars, the edges that can be the ending locations for cars (this can be set to all edges if required), how many cars per tick, and tick size.

This can be used to simulate real life events where many cars leave from the same place at roughly the same time.

2. Incoming Event

An incoming event can be used to simulate events where many people cars arrive at the same place at roughly the same time. This script responsible for this works by estimating how much time it will take cars to reach that location, and creates cars, such that they arrive at similar times. The estimation for how long each car will take to reach the location is calculated by first applying Dijkstra’s algorithm and finding the shortest path. It then calculates the time it takes for a car to go through that street by dividing its length by its speed limit. After that it sums up the time it takes to go through all streets on the route, and lastly, it subtracts the time it will take from the desired time of arrival.

Using the Traffic Generator

Remark
The user has to decide which data generation method he wants to use. It is possible to use only one or all four depending on the scenario. If you wish to use flows only you can skip to point 2, as edge probabilities are not required for flows. If you wish to use real data only, you can skip to point 3. If you wish to use event generation only, you can skip to point 4.

1. Generating Cars According to Edge Probabilities

1. Setting Edge Probabilities

First, the user has to select whether he wants to use a random distribution or a uniform distribution as a starting setting for the probabilities.

The uniform distribution function (**uniform_prob**) takes as input:

- prob – the desired probability that will be set for all edges
- edges – the number of edges

The random distribution function (**random_prob**) takes as input only the number of edges.

After running one of the two functions, the output is edge_start_prob and edge_end_prob. These are lists containing how likely an edge is going to be a starting position for a car or an ending position of a car. The index of the lists serves as the edge number.

The data can then be modified for the required purpose. There is a function to edit the data called - **edit_edge_prob**. It takes as input:

- edge - the edge number, it can also be a list of edges
- start_or_end – whether you want to edit the starting or ending probabilities

2. Generating Cars after Setting Probabilities

After creating the probabilities, we can move on to creating the cars or blocks. The car_gen_start is the function that is responsible for creating cars according to the starting probabilities we have created earlier (edge_start_prob list).

The input for the function **car_gen_start** is:

- Maxtime – the time at which you want the generator to stop creating new cars
- Start_point - the starting points – which are set to 0.5 by default (the center of the edge), and end points – also set to 0.5 by default.

This function creates a car list of the generated cars. The car list contains the starting edge, starting point, ending edge, ending point, starting time.

The other function **car_gen_end** creates cars according to end probabilities (edge_end_prob). The inputs are the same, and the output is the same.

3. Editing the Car List

The function edit_car_list is used to edit properties of cars. It can be used to change the starting edge, the starting point, the ending edge, the ending point or the starting time. **edit_car_list** takes as input:

- Car – the index of the car whose properties are to be changed
- Prop – 0/ 1/ 2/ 3/ 4, where the number corresponds to: starting edge/ starting point/ ending edge/ ending point/ starting time
- New_value – the new value for the selected property

2. Generating Cars Using Flows

1. Timed Flows

Timed flows can be implemented by using the function **flow**. Its inputs are:

- starting edge and ending edge
- starting point and ending point – both are set to the middle of the edge by default
- start_time and end_time – the beginning and ending of the flow
- tick – the tick size, set to 1 by default (a new car is generated at every heartbeat)
  The output has the same format as the output of the previous generator, a list containing: starting edge, starting point, ending edge, ending point, starting time.

2. Flows with Probability

The flow with probability **flow_prob** has the same inputs, the only difference is that there is one more value to be input:

- prob - the probability - how likely a car is to appear at each tick

3. Generating Cars Using Real Data

The function **syn_data_gen** can be used in order to generate cars based on real data. The function has the following inputs:

- File_name – the name of the file containing the real data. It has to be in the same format as the outputs of the generator. It has to have a column 0 – containing the indexes of cars, a column 1 – starting edges, a column 2 – starting points, a column 3 – ending edges, a column 4 – ending points, a column 5 – starting time.
  It is also possible to exclude columns 2 and 4 if the real data does not contain points and edges only.
- Sample_size – the required amount of cars that the user wants to generate
- Model – CTGAN/ GaussianCopula/ CopulaGAN/ TVAE – the required model

4. Generating Cars for Events

1. Outgoing Event

The outgoing event function **out_event** takes as input:

- Start_edge, start_point

- End_edges - it has to be a list with square brackets [] (a 1-d array), even if it is one edge. It does not work correctly if the input is not in square brackets, due to Numpy's random choice function used in the script.
- ending point – set to 0.5 by default. It is not possible to generate cars using this function that end in different points. This however can be done by using the edit_car_list function (see 1.3)
- start time, end time – time diapason in which cars are generated

tick size and cars per tick

2. Incoming Event

The incoming event function **in_event** takes as input:

- Start_edges – Has to be a 1-d array
- Start_points, end_point, end_edge
- End_time – the function does not have input for start time, as that is unknown and has to be calculated by the script
- Cars – the desired number of cars

5. Clear Car List

The function **clear_car_list** takes no inputs. It is used to clear the global variable car_list. It can be used whenever the user wants to create a new list of cars.

Limitations

A limitation of the traffic generator is that in order for it to create synthetic data based on real data it needs to have complete training data. It requires a list of cars – their origin, their destination and their departure time. SUMO on the other hand requires less in order for it to imitate real data. SUMO is capable of generating traffic based on data coming from counting devices. It can create realistic traffic only from information about how many cars pass through a specific edge.

Another limitation is that probabilities are static. Both in the generator based on edge probabilities and in the flow with probabilities. This means that the probabilities cannot change during the data generation process. Once they are set, they remain the same during the whole generation. This is something that SUMO is capable of doing as their flows probability can change dynamically according to a distribution function.

Conclusion

The Traffic Generator allows for using the simulator to its fullest potential. By generating data using one or more of the four tools, the user can create a list of cars that can then be used in the simulator to achieve different purposes. We have successfully reached out goal to create possibilities for generating cars for all scenarios that might be of interest. The user can create cars that start from specified edges and end in other specified edges. Flows can be used in scenarios where the user requires a car followed by other cars. Real data can also be used to create realistic synthetic data. Events, where multiple cars end in a single location or start from a single location, can also be simulated using the event generator.

References

Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. "Microscopic Traffic Simulation using SUMO", IEEE Intelligent Transportation Systems Conference (ITSC), 2018.

“The Synthetic Data Vault”, sdv.dev, accessed 2022-07-10

Zhang, Huichu, et al. “CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario.” ArXiv.org, 13 May 2019, https://arxiv.org/abs/1905.05217.