SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.107

Published: 29 Jun. 2022

A comparison of SUMO's count based and countless demand generation tools

Michael Behrisch[1] and Pauline Hartwig[2]

[1] German Aerospace Center, Berlin, Germany
michael.behrisch@dlr.de
[2] TH Wildau (University of the Applied Sciences), Wildau, Germany
paha2034@th-wildau.de

Abstract

There are already several tools available to generate traffic demand for the microscopic simulation suite SUMO. This paper focuses on setting up a simulation scenario for the peak hour in a small conurbation when there are vehicle counts available for the major streets. We describe tools which are part of SUMO or available as open source and compare their results with the real traffic counts as well as with the outcome of countless demand generation.

1 Introduction

Setting up a microscopic traffic simulation scenario is hard work. While it is easy to get started with a network derived from OpenStreetMap, fixing the network to fit microsimulation purposes and getting a proper traffic demand are two tasks which can consume considerable amounts of time. Fortunately there are tools to support both tasks which can ease the work. While the network adaption is still largely a manual process supported only by better editing facilities in tools such as netedit, traffic demand generation can be done in an automated way and still deliver good quality results. This paper focusses on several demand generation tools usable with the SUMO simulation suite [2]. It does not cover "traditional approaches" such as setting up an origin destination matrix, generating trips from there and then doing traffic assignment. These tools are described in detail in the available literature for an overview see [3].

We instead focus on the quick (and sometimes dirty) solutions to give people an immediate but still somehow realistic travel demand for their area of interest assuming no additional input beyond the OpenStreetMap network and a few counting locations (which are even optional for some of the tools). The motivation is to replace as much of the manual work as possible why still generating plausible routes, realistic counts and a good coverage of the network.

The paper is structured as follows. After describing the solution approaches of the various tools with and without counting input and give a brief account on their strengths and weaknesses, we compare their results to the counting data we used as input and two additional counting locations not used as input. Furthermore we address the problem of routes primally running on the main network by evaluating the network coverage. We end with conclusions and an outlook for further research.

The underlying scripts and results are all publicly available in the sumo scenario github repository [1].

2 The scenario

The tools we evaluate need a realistic yet small use case such that all basic features can be demonstrated but it is still feasible to do manual adaptions to the network and evaluate the data if needed. For these reasons (and also because of geographic proximity) we used the network of the town Wildau (Germany, Brandenburg), which was extracted from OpenStreetMap using the OSM WebWizard. Wildau is a small town in Brandenburg/Germany which has around 10000 inhabitants and a surface of 909 ha. The generated cutout as visible in Figure 1 includes a total edge length of 104.06 km and 645 nodes and 1426 edges.

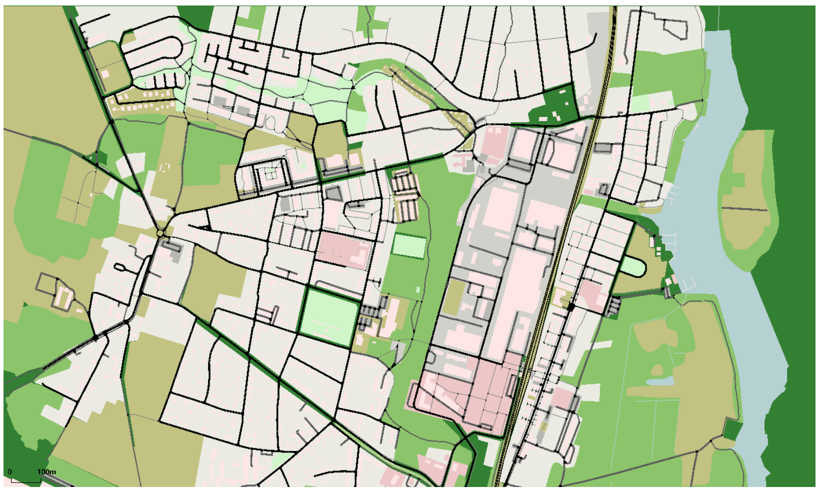

Figure 1: SUMO scenario

After the automated extraction from OSM the next step was to clean up the network mostly for faulty or missing lane to lane connections at the junctions. This is a necessary step in most simulation setups. If there is no knowledge of the local situation the easiest way to find the wrong connections is to generate a large random demand and look for the junctions which jam first. This already gives a usable network for our purposes. Since we strived for more realism, the maximum speed and the allowed vehicle classes have been adapted too (especially concerning bicycles).

The generated demand only covers the afternoon peak hour (15:00 to 16:00). The reference data for the number of (passenger) cars has been collected at nine counting points provided by local authorites and one manual counting point as visible in Figure **??**. The initial approach (and baseline for our studies) was a manually generated flow file, which includes routes based on assumptions about the local demand situation. The number of vehicles using the different routes had been scaled using Excel based optimization tools to adapt them to the counts. This resulted in a total demand of 2254 vehicles in the initial route set.

3 Countless tools

The easiest way to generate a demand would be if the user needed no extra data or could at least enter just a single number as the expected population or the total amount of vehicles per hour. We call these tools *countless* and evaluated them first for two reasons. First they are obviously easier to use and second while their output is not perfect it may still serve as a base for optimization in further steps.

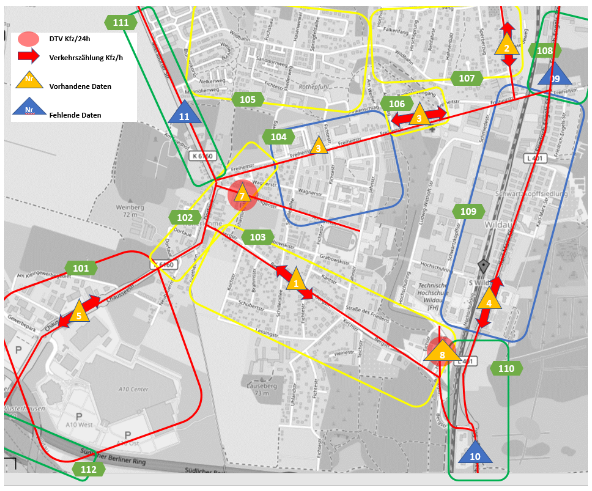

Figure 2: Counting locations

3.1 randomTrips.py

The tool randomTrips.py is part of the core SUMO suite and probably for most people the first contact with demand generation, because it also works behind the scenes in the osmWebWizard tool. As the name says it randomly generates trips (origin destination pairs of edges) and with the right options also completes them to full routes and validates them for drivability (mostly connectivity) within a network.

The selection process can be tailored heavily to prefer edges with more lanes, higher speeds or greater length but in our experiments we stuck to the defaults for most of these values. The subsequent route generation is done using a hidden call to the duarouter which runs a shortest path calculation to find the route set and discards all disconnected pairs of origin and destination.

To adapt the traffic volume two major parameters have been changed, first the fringe factor (increases the probability that journeys will end / start at a boundary edge), and the period (how often new vehicles are generated). Already the first runs showed that randomTrips is able to generate a better network coverage than the manual routes which only focussed on counting locations. This is not surprising since a random choice will obviously cover more than just the major streets.

Subsequently, the value of the "period" was reduced in order to generate more traffic. The procedure was repeated until the first heavy jams occured at the junction Bergstrasse / Dorfaue. In the second step, the boundary edges were examined, which initially showed a low level of utilization. In order to be able to adapt this, the "fringe factor" was increased. The value was increased until there were no more vehicles teleported in combination of the "period". In the end, with the values $f = 7$ and $p = 1.850$, a relatively good distribution picture of the traffic was generated.

Using the option "random" (to modify the random seed and hence the sequence of generated start and end points) several different demands have been generated to get an average result for the evaluation. To calculate the total counts on the edges with counting locations the meanData mechanism of SUMO has been used which gives aggregated data for the whole network.

3.2 SAGA

SumoActivityGen (SAGA) is a tool which takes only the extracted OpenStreetMap file as input and tries to generate the complete demand automatically. Public transport, motorized vehicles and pedestrians are generated here. The time span comprises a complete daily cycle. The route generation is based on possible work routes. The tool "looks" where workplaces, houses, etc. are located and determines the routes based on this information from OSM. The tool was used on Wildau, however the "raw" network was used imported by OSM WebWizard. To let SAGA generate any demand at all for the small network we decided to start it with the "single-taz" option because it otherwise tries to generate demand only between districts (in the case of Wildau between the city itself and the surrounding ones) but not inside the district.

To calculate the total counts on the edges with counting locations we again used the meanData mechanism of SUMO focussing on the peak afternoon hour. The total number of vehicles generated by SAGA was much less than in the manual demand file, details will be dicussed in section 5.

3.3 randomActivitygen

A tool similar to SAGA is "randomActivityGen". It serves as a randomizing wrapper to "activitygen" which is part of the core SUMO suite. It takes the SUMO network and the number of inhabitants as input and generates random work and home locations to induce a traffic demand.

When evaluating this tool, it became apparent that the number of vehicles generated was very low. The number of inhabitants was initially given as 10000, which is close to real population. whereby the number was increased to 50,000 when called up again. The number of vehicles increased slightly, but the traffic generated was still implausible. This is probably due to the fact that the tool does not take into account surrounding cities / factors. Due to its close location to Berlin, Wildau has a high proportion of through traffic / commuter traffic, which means that the traffic flows would have to be correspondingly higher.

4 Tools using traffic counts

4.1 dfrouter

In addition to the tools already used, other methods were also used. First, the "dfrouter" was implemented, which receives defined (induction) loops with the relevant edge IDs and a vehicle file in semicolon-separated format with the number of vehicles as input. The basic idea behind this tool is that incoming and outgoing flows can be recorded almost completely using induction loops. With the help of the information, the dfrouter rebuilds the number of vehicles and routes. This is done in four processing steps. First of all, it must be specified in the loop file whether it is a source, target or intermediate detector. In the next step, routes between the detectors are calculated, whereupon the flow rates are generated. To generate the flows dfrouter starts at every source with the demand measured there and distributes it according to the downstream detectors untila sink detector is reached.

Except for determining manually which detectors are considered sources and sinks there is no easy way to influence the behavior of the process so the results are basically stemming from a straight invocation of the tool. The dfrouter generated the routes and vehicles as output files and the result has been evaluated again by running them through a scenario and measuring edge usage using meandata files.

4.2 flowrouter

In a second method, the "flowrouter" was used, which works in a similar way as the "dfrouter". The flow router tries to find a number of routes that maximize the overall flow in the graph theoretic sense. So it treats the measuremnts as capacities for the edges and maximises the number of vehicles in the network which travel from source to sink edges. The input remained the same, but instead of individual routes, "flows" are generated for the vehicles. The traffic flows received were evaluated using a further output file and transferred to Excel. It must be mentioned here that both tools mainly use the edges covered by detectors (and the ones connecting them) and do not generate any traffic into the secondary network.

4.3 cadyts

Another option for route selection within a network, which is based on counting data and route alternatives, is offered by the "cadyts" tool. Cadyts performs 100 iteration steps in which the probabilities of the route selection from the route files are tried to match the counting data. For this purpose, the results of the previous iteration are evaluated in each iteration step and the probabilities are adjusted if necessary, in order to get close to the counting data. In order to be able to execute cadyts, however, duaIterate must be executed first. DuaIterate creates an alternative route file for each iteration (the number of iterations depends on the time slot), which is then passed on to cadyts. There are a variety of parameters to be adjusted when using duaIterate as well as cadyts to influence the calculated user equilibrium. Due to the limited time the calibration work could not be finished and as a result the deviations were still very high.

4.4 routesampler

In a further attempt, the "routesampler" was used to assign the counting data to the corresponding edges so that the randomly selected routes match the counting data. In contrast to the other tools the routesampler already receives a route file as input together with the usual "counting data". Its algorithm chooses iterativela routes from the route file to match the counting data. While the usage of an exisiting route file sounds like a severe limitation it is possible to use the output file from "randomtrips" as input for the model and still get proper results. This route file can be used together with a counting file in the meandata format (which has been generated manually) to start the calibration. By calling up the changed batch file again (now with the "routesampler"), the new scenario was generated and could be evaluated using a new output file. The result was that the routesampler can match the counting data, which were given as input.

5 Results

The results were evaluated with respect to two different indicators which are the precision of the fit (denoted by the root mean squared error of the measured counts in relation to the simulated values) dureing the peak hour and the coverage of the network (total length of streets used by simulated cars divided by overall length of car edges in the network).

With the help of the root mean square it became clear that the routesampler was the most suitable tool to achieve the counting data and a plausible route distribution. A python script was created for the evaluation of the individual tools, which calculates the root mean square of the counting data. Here, the count data were considered with the output data of the edges and the quadratic deviation was calculated. In the end, the sum of the squared deviations was calculated. The lower the value, the better the tools were. During the evaluation the routesampler and the manual route data achieved the lowest values, which means that these methods were the best during the test. On the other hand, SAGA and randomActivityGen were not suitable for generating demand due to their high values. However, the value of randomActivityGen with 50.000 inhabitants was slightly lower than 10.000 inhabitants.

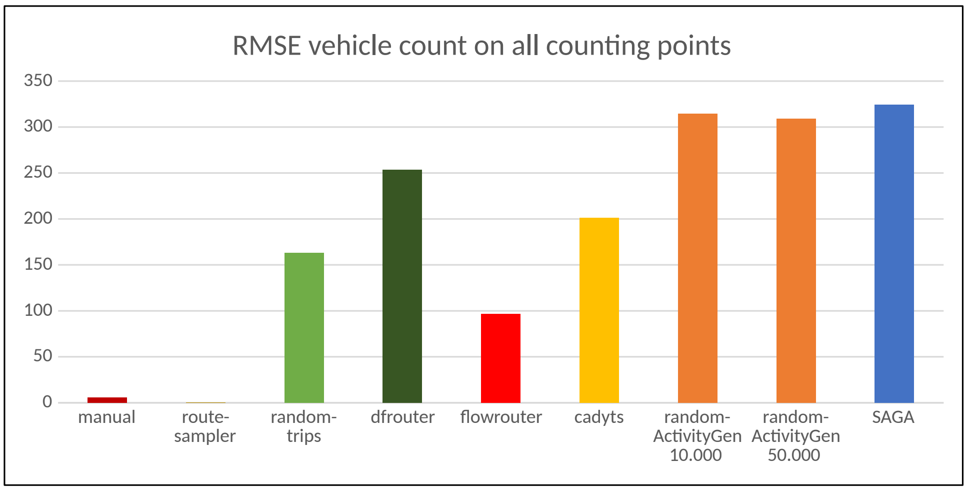

Figure 3: Root mean squared error of the vehicle counts

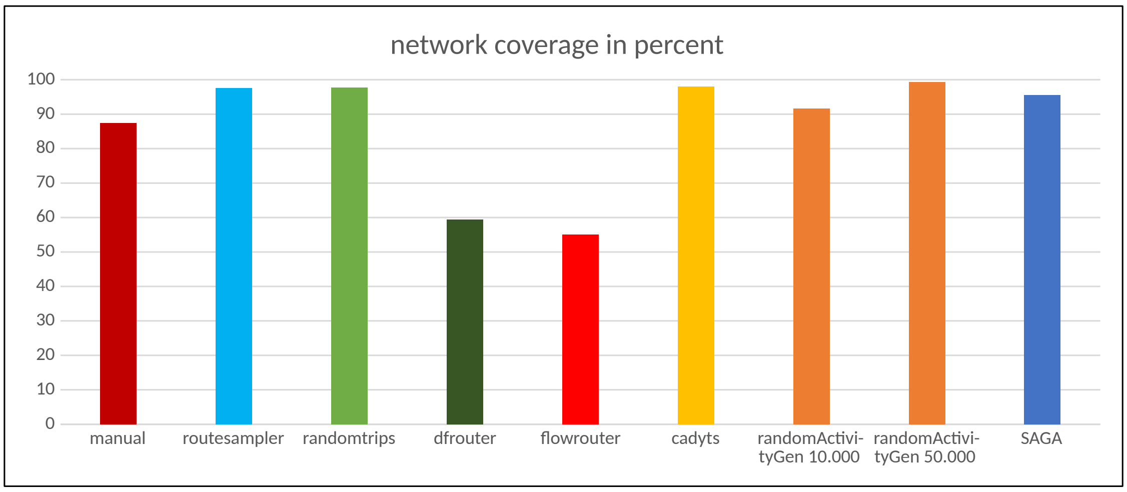

Figure 4: Percentage of the street network covered by vehicle traffic

The conclusion is that the routesampler is the best way to simulate the demand however the route distributions (main and secondary network) are balanced.

6 Future Work

In the future it is planned to expand the simulation on github. On the one hand with the local public transport and on the other hand with the bicycle and pedestrian traffic. Furthermore, the presented tools are to be further tested and evaluated with the help of this paper. Here it is conceivable to change the parameters, in which it is possible to find better results. The paper is an invitation to work with the tools interactively. Maybe some users find other methods to work with them or have some ideas for new definitions for parameters.

Furthermore the results should be used to foster automatic creation of realistic scenarios. The manual steps which were still necessary for instance when calibrating the randomTrips scenario could be partly automated at least the step of increasing the traffic until the first teleports (i.e. heavy jams) occur. It is probably not possible to automate the choice of the fringe factor though because it deponeds more onthe surrounding network than on the chosen network itself.

Last but not least further insights in the process of the user equilibrium would allow to do a joined process between duaIterate/cadyts and the routeSampler or do a better comparison of both tools and processes in the future.

7 Acknowledgments

We want to thank the city and the administration of Wildau as well as the University for providing the necessary data to perform this study. Furthermore this project has been initiated by a student project together with Ewald Bader and Andesson Wafo who contributed to the first scenario.

References

[1] SUMO scenarios. https://github.com/DLR-TS/sumo-scenarios. Accessed: 2022-06-17.

[2] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[3] Dieter Lohse and Werner Schnabel. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 2-Verkehrsplanung*. Beuth Verlag, 2011.