Online Micro Modelling Using Proprietary Controllers and SUMO

Robbin Blokpoel and Jaap Vreeswijk

Abstract Over the past years the open source traffic simulator SUMO has been significantly improved and extended. One of the most important elements of urban traffic simulation is the proper handling of traffic light control. Currently available are elementary control methods like embedded fixed time and actuated control, but also controllers external to SUMO that use SUMO's extensive TraCI interface that enables reading and changing of many simulation parameters. This interface, however, has as yet not been used to link to proprietary controllers, which would enable the use of SUMO for accurate studies in a multivendor environment. Moreover, the TraCI interface accepts the injection of vehicles from external sources during the simulation. This opens up possibilities for using real-world sensor data directly in the simulation environment. This paper describes how state-of-the-art Imtech controllers are linked to SUMO. The paper covers topics like architecture, vehicle detection, signal group control, simulation speed optimization and contains a comparison of the SUMO simulation to the commercial Vissim simulator for an identical scenario. The last section of this paper introduces embedded real-time micro simulation as part of the control environment, which was able to approach.

1 Introduction

Over the past years the open source traffic simulator SUMO has been improved and extended significantly with at the time of writing a 19th version available. With a large community involved and a history of more than 10 years, the simulator can be

R. Blokpoel (✉) · J. Vreeswijk
Imtech Traffic & Infra, 2542, 3800GB Amersfoort, The Netherlands
e-mail: robbin.blokpoel@imtech.com

J. Vreeswijk
e-mail: jaap.vreeswijk@imtech.com

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_3

considered a serious alternative to commercially available solutions. The open source nature and easy access to almost all parameters during runtime make the simulator particularly suitable for research projects. Therefore, the European funded project COLOMBO [1] chose to use SUMO for traffic simulations.

The COLOMBO project works on traffic surveillance algorithms for low penetration cooperative systems [4], in which both vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications are modelled using the ns-3 [7] communications simulator. The output of these traffic surveillance algorithms is used by new traffic control algorithms to control signalized intersections.

Currently SUMO supports various kinds of traffic control; fixed time and vehicle actuated control are fully supported by SUMO. For other types of control and variations on the embedded vehicle actuated method, external controllers can take over SUMO's control through TraCI (Traffic Control Interface). Currently, these external controllers, like the example Python program that comes with SUMO, are stand-alone applications specifically made for connection to SUMO. However, for a good comparison between traffic systems currently running on the street and results from research projects, it is important to use the same scenario and simulation environment. Therefore, an interface between SUMO and a real-world controller would be a very useful tool for COLOMBO to compare its solutions to what is currently available on the market. This comparison would be even stronger when real time or historical traffic demands over a long period of time can be fed to the system. That would prove the system could work over an extended period of time and not just in one scenario.

Fixed time and vehicle actuated controllers can be realistically approximated by either SUMOs internal traffic control options or external applications. City specific rules about pre-starts, not early cutoff and variable safety margins according to detection information can make this a very complicated task that favour using the real world controllers. This holds even stronger for traffic adaptive control, like Imflow [9], which is too complicated to be approximated by the control available in Sumo. Furthermore, the differences between competing products are too large to simulate their behavior with a reference application.

For these reasons it was decided to create an interface between Imtech's real world controllers and SUMO. This paper describes the architecture, detection, signal groups, speeding up the simulation and a comparison between Vissim and SUMO. This is done for the scenario of Assen-Noord, a small network in the north of the Dutch city Assen. Network conversion between Imflow controlled networks and SUMO for increased ease of use is described in [2].

Additionally, this interface enables to use the simulation environment as accurate online model that only requires traffic demand data. Unlike many other models [3] no data about travel times is required. Exceptional changes in demand, which are often a problem for traffic models based on travel time measurements and neural networks [7], are handled better. This is because the real traffic light controllers are part of the model and behave the same way as the controllers on the street would behave. The paper shows a test case with a week of data from a network in Helmond in the Netherlands.

2 Architecture

The architecture of the interface and all involved components is described in the picture (Fig. 1).

The TLC (Traffic Light Controller) blocks in the diagram use the same software as is running inside real-world traffic light controller. The TLC blocks can accept just as easily real sensor input as data generated by the simulation environment. The equivalent situation holds for the actuators. The TLC interfaces to the simulation environment by means of the SimInterface. The SimInterface is a C++ dynamic link library (dll) that can maintain connections to multiple TLCs in parallel. On the other side it can connect to any external application that supports the dll. This has been used to connect to the commercial simulators Vissim, Paramics and Aimsun. For SUMO an intermediate block, the SumoInterface, has been created in Java that supports both the dll and can talk to TraCI to get information from SUMO. The flow of information consists of two main flows, detection information going from SUMO to the TLCs and signal group status from the TLCs to SUMO.

After initialization of the interfaces the main interface process checks detector status and signal group status every 100 ms. Changes to signal group status are written to SUMO, while detector status is sent directly to the Siminterface dll. Finally, SUMO is ordered to execute another simulation step through TraCI.

3 Detection

Detection—the acquisition of traffic sensor data—is a key element for any adaptive controller, as without detection only fixed time control is possible. Therefore, having proper detection functionality is vital for accurate simulation. SUMO supports three kinds of detectors: inductive loop, lane area and multi-entry multi-exit detectors. Current traffic control is mostly based on detectors that cover an area in a lane, this can be an inductive loop, but also a marked area in a video detector. Therefore, the original inductive loop detector of SUMO is actually not sufficient

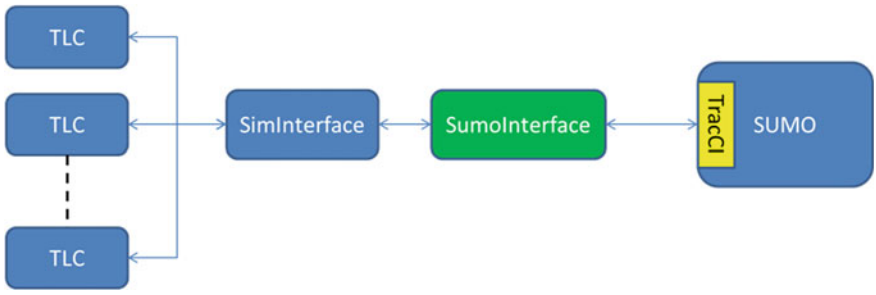

Fig. 1 Interface architecture

for traffic control simulation, since it's an infinitely small detector that doesn't cover an area in a lane, but just a point on the lane. Even real world inductive loops cover larger areas, so a real inductive loop cannot be modelled accurately with a SUMO inductive loop. Many vehicle actuated strategies use long induction loop area detectors that can cover up to 30 m of the approach to an intersection. By means of these long detectors gaps in the approaching traffic can be detected, which can be used to determine the best moment to cut off the green phase.

Figure 2 shows a short loop close to the stopline and a 20 m loopat some 15 m in the upstream direction. When a vehicle leaves the long loop, the front of the vehicle is at a distance of at most 12 m from the stopline. Turning the light to amber at this moment will not make the vehicle stop, since the distance is less than 1 s. This enables the controller to utilize part of the amber time by letting the last vehicle of a platoon pass through during amber. This technique of detecting the end of the platoon would also work at the stopline, but then the amber time cannot be utilized. The reason for using a long loop of 20 m is to deal with small gaps in platoons due to different acceleration rates. If the loop would be shorter, a threshold gap time would have to be introduced that would make usage of the amber time impossible. The most usable alternative would be a small loop at 15 + 20 = 35 m from the stopline, but its efficacy would depend on a presumed fixed vehicle speed, which is not realistic close to an intersection. Moreover, this work focusses on simulating real world controllers and these expect long area loops. Using different loop configurations in the simulator will give different behavior unless parameters inside the controllers are changed.

Interfacing with the detectors through SUMO is quite straightforward. During the development of the interface a small extension to Traci was made to be able to access occupancy of lane area detectors. This extension is available in version 0.20. This is done using the command "get LaneAreaDetector Variable" and the variable to acquire the number of vehicles on the loop. In the dll this is fed back as a list of detectors that can be occupied (1) or not occupied (0). Main challenge in this is the configuration, since the dll does not use detector IDs. The order of the detectors has to be the same as it is configured in the controller executable. This problem was previously solved for Vissim simulations by a naming convention, detector numbers

Fig. 2 Typical detection field for vehicle actuated control

have an ID number specified as follows: intersection ID * 1000 + detector sequence number. So the first detector for intersection 37 has an ID of 37000, the second 37001, etc. The network conversion tool of [5] automatically uses the correct naming conventions when the original network uses the correct numbering scheme.

As described in the architecture section, the update time for detectors is 100 ms. This is done in order to never miss any detection event. Motorcycles can be as short as 2 m and on the highway, their speed can be over 30 m per second. This means they occupy a detector for only 100 ms. When vehicles are shorter and drive faster, a shorter update time is required. In urban environments the simulation may be speed up by checking the detectors less frequently if vehicle speeds are lower. For 4 m vehicles at 15 m/s, a 300 ms cycle suffices.

Another important aspect to consider is the stopping distance in front of a red light. This is shown in the figure (Fig. 3).

The loop indicated by the blue line is not occupied when vehicles stop at 2.5 m before the signal head, as they did in older SUMO versions. Therefore, the request is not registered at the traffic light controller and the signal group will never become green. In SUMO 0.20.0 this stopping distance was decreased to 1.0 m.

4 Signal Groups

Sumo uses a different kind of numbering for the signal groups than is usual in traffic light controllers. Vissim has one signal head per lane and a signal group can comprise multiple signal heads, which happens for example when there are two lanes for a certain direction. SUMO, on the other hands defines connections, which can be considered signal heads, in the .net.xml. There is one connection per turn direction per lane. So when there is one lane from which a right turn, the through direction, a left turn and a u-turn are possible, it will have 4 signal heads as opposed to only 1 in Vissim. Therefore a translation XML file is used by the interface to convert TLC signal group numbers to SUMO identification numbers. An example of a translation file is shown below:

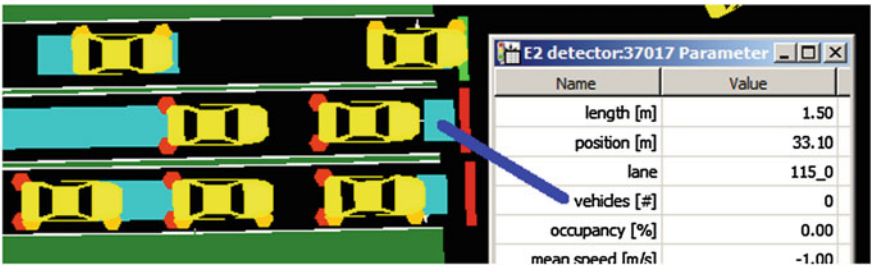

Fig. 3 Problem with detector location and stopping distance

```
<intersection id = ``1" >
<signalgroup id = ``1000" sumoSGs = ``2, 3, 4"/ >
<signalgroup id = ``1001" sumoSGs = ``5, 6, 7"/ >
<signalgroup id = ``1002"sumoSGs = ``1, 8"/ >
</intersection >
```

The translation is made as an add-on to the software of [5] during the network conversion process. Per edge the convertor knows which Imflow signal group number belongs to it, while the list of connectors per edge in the .net.xml is also known. The conversion file simply contains per signal group ID, the list of linkIndices. During operation of the interface the traffic light status is translated according to the file. Suppose the controller wants signal group 1000 green and the rest red, the SUMO translation is as follows: rGGGrrrr.

Again in the dll there is no ID registration, the order is always the same and therefore it is important that the translation file has the signal groups numbered according to the order in which they are configured in the controller executable. Also, there are more states defined than in SUMO: undefined, green, red, off, red +amber, amber, amber flashing, red flashing, green flashing, red+green flashing and green+amber. Some of these states don't exist in SUMO and are converted to simpler states, like red+amber is functionally red, so it will just show red in SUMO, since the driver model wouldn't take this into account. Similarly, green+amber is just shown as green. Most flashing states are implemented to show "O" for half a second and "Y" or "G" for the other half second. Note that the symbols have to be capital otherwise vehicles may decelerate unnecessarily. For red flashing it is slightly different, it will just show continuous red to prevent vehicles from entering the intersection while the light is temporally off as part of the flashing. When no external controller is connected to the dll, the state is automatically set to amber flashing. During operation in every 100 ms the software checks whether the status has changed and if so sends a "Change Traffic Lights State" command with a new state tuple String. The reason to choose for new state tuples is because the traffic light can show many combinations of some lights being yellow while others are still green during stage transitions. Putting all these possible combinations into either a program or predefine them in a SUMO configuration file and selecting the right phase index during operation would be a lot of work.

5 Simulation Speed

It was noticed that the network used for testing the SUMO interface between TraCI and the SimInterface was running much slower after the detectors were connected. Although the number of detectors is high, with 168 divided over 5 intersections, the delay was much larger than expected. An implementation that sends separate TraCI commands for each detector requires up to 30 ms per intersection per simulation

step of 100 ms. This meant that the simulation ran approximately at the same speed as real-time speed (on a 2.53 GHz core 2 duo). So each second of simulation took one second on the clock. Without detection this speed was 50x real-time. A hypothesis that the large number of Traci calls caused this led to combining all detector requests of one intersection in one call. This led to an increase in simulation speed to almost 2x real-time speed, which is an improvement with respect to the first implementation, but still not acceptable. It appears there is an internal SUMO problem with TraCI causing the large delays. Subscriptions are also not going to solve this problem, because reducing the number of requests from 168 per timestep to 5 did only marginally decrease the delay. A further reduction from 5 to 0 would not reduce the delay significantly. Further investigation in cooperation with the SUMO development team is required to investigate this issue.

6 Comparison Between Vissim and SUMO

For the scenario of Assen-Noord, a small network in the north of the Dutch city Assen, a comparison was made between VISSIM and SUMO. The network only has pedestrian and bicycle crossings at the middle left intersection. All other intersections have just vehicles. The larger traffic streams (up to 1500 vehicles per hour) are going north–south on both sides of the network and the major bottleneck is the bottom intersection where the two north–south streams join (Fig. 4)

When watching the simulations in both Vissim and SUMO, no clear differences could be noted, except that SUMO has uniform vehicle injection and the same acceleration at the stopline. SUMO was used in a standard way creating the routes with a trip file that injects vehicles with a constant time period in the resulting.rou.xml. Evaluation in Vissim was done by putting a travel time section for each signal group and in SUMO a multi-entry multi-exit detector.

When evaluating the results it was found that the vehicle count in SUMO was off, sometimes only 35 % of the actual volume was measured. It appears to occur mostly when there is a high density on the multi entry multi exit detector, since signal groups with low volume were counted correctly. The delay time could be acquired directly in Vissim, but in SUMO a run with all signal heads switched to "O" was done to acquire the free flow travel time, which was subtracted from the measured travel time to get the delay time. From this it could be noticed that on average the delay for pedestrians and bicycles was 2.0 s higher for SUMO than for Vissim. On the other hand, for normal vehicles this delay was 1.3 s lower for SUMO.

These results were obtained using standard settings as much as possible. However, SUMO has many options for car following models and different vehicle models with other acceleration parameters for vehicles, bicycles and pedestrians. Tooling also exists for more random vehicle injections with normal or Poisson distributions. Using these options will make it possible to have the results closer to the Vissim simulation results.

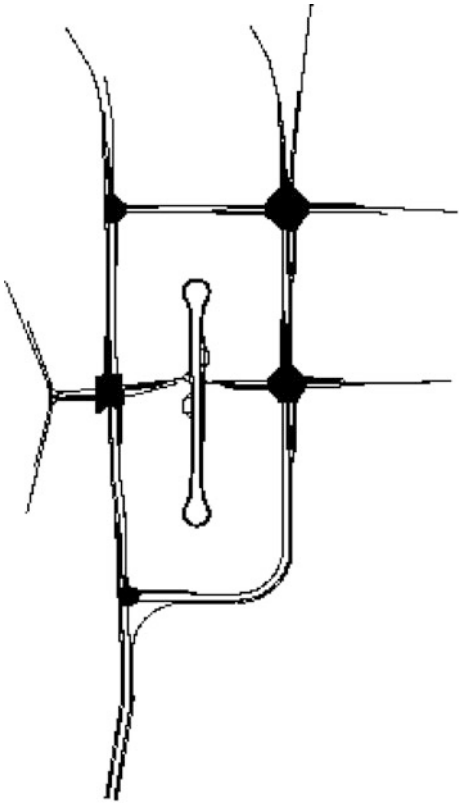

Fig. 4 Simulation network of Assen-Noord

7 Online Micromodelling

For accurate online micromodelling there are three main components that should be considered: traffic behaviour, traffic demand and traffic light control. The latter is covered by connecting the real traffic light controllers to the simulation. Traffic behaviour was found to be close to Vissim, which is generally considered an accurate model [10]. Therefore, the only remaining component is the traffic demand.

Generally, traffic counts are easily available, since traffic light controllers require counting sensors, like inductive loops to function properly. A lot of research has already been carried out for estimating OD matrices from loop counts [6, 7]. The method used in this paper only uses traffic counts, but no vehicle class specific counts. A method with class specific counts is presented in [8]. For this research the formula taken from [6] will be used:

$$\sum_{w \in W} p_w^a t_w = v_a$$

t_w is the number of trips of O-D pair w,w∈W
p_w^a is the proportion of trips O-D pair w∈W traversing link a ∈ A
v_a is the expected link flow for the link a ∈ A

There are, however, multiple valid solutions for this equation and therefore some assumptions are needed. The flow per origin can be measured directly since there are inductive loops at each entry of the simulation network under investigation, but for destinations there are multiple possibilities.

This is best understood when considering Fig. 5; a vehicle entering the network at intersection 102 going in the eastern direction can leave the network at 104 in three different directions. The same holds for a vehicle that entered at 101. Since these vehicles mix with each other inside platoons, they cannot be distinguished anymore at the loops of intersection 104. This also demonstrates that this knowledge is not necessary, because of the same mixing effects. Only in extreme cases the effects of different destination ratios from different origins will be noticable. For example when all vehicles entering the network at 102 turn left at 104, while none of 101 and 103 do that, an unexpected gap in a platoon may occur because those vehicles will still be relatively close together inside the platoon. Similarly, at the first intersection after entering the network the vehicles of a certain specific origin will also arrive at a specific time. In this case it is important to have a more specific indication in which direction they proceed, since this is important to model the effects of coordination between intersections correctly.

The solution for determining the destination in the online model is to use the ratios from a detailed static OD model for the first intersection a vehicle encounters and afterwards group traffic to follow general turning percentages at the following intersections. This detailed model was made to accurately represent the traffic demand of the network in agreement with the road operator. The resulting flows per OD pair are used as the basis of a Poisson arrival process, which injects vehicles into SUMO through the TraCI interface.

The data of the traffic counts also contains total red phase durations, which should approximate the total red duration of the online model environment. This was tested for 2 weeks of data in March 2014. As a comparison the actual total red time duration of each signal group was compared between the actual street data and

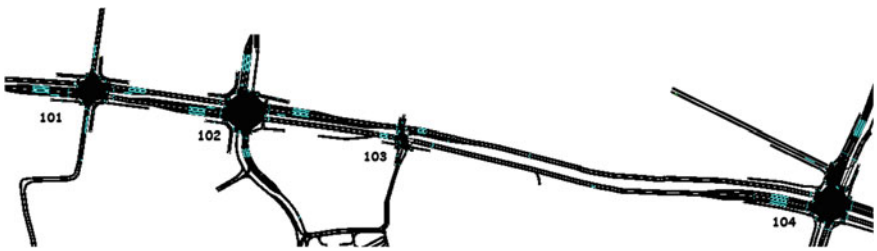

Fig. 5 Simulation network of Helmond for online modelling

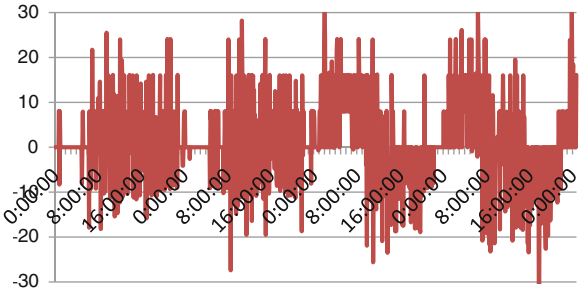

Fig. 6 Difference between simulation and actual street data from 7–10th of March 2014

the data from the simulation. The resulting difference in total red duration per 5 min for signal group 3 (left turn from the main direction) at intersection 104 is shown in the figure (Fig. 6).

As can be seen from the figure, the difference between the actual data and the simulated data is very irregular on the first 2 days. This can be explained by sampling effects, a green phase that occurred at the border of an interval can be either in the first or second interval. Moreover, the Poisson arrival process introduced some randomness in the demand as well, which could easily lead a difference of one green phase per sampling period. When data is aggregated over 30 min, the deviation does not vary between −15 and +15, with exceptions up to 30 anymore, but only increases to −30 to +30 with exceptions up to 60. This effectively doubles the variation, while the interval length is increased by a factor of 6. However, the last 2 days have a clear pattern on the difference. From midnight to midday the simulation has less red time, while at PM hours it is the other way around. A deeper investigation into the log files revealed that the controller switched to another plan at the 9th of March at 0:50AM.

When looking at the average deviation for signal group 3 of 1.5 s, it can be concluded that the simulation was quite accurate. The average absolute error was 4.9 s, but this includes the noise introduced by the Poisson arrival process and the sampling effects. For all intersections the average error and average absolute error over all signal groups were calculated and are shown in Table 1. The average absolute error is partially due to deviations from the Poisson arrival process. When putting these errors of 0.0 and 8.9 s in perspective with the data aggregation period of 300 s, the average total error is <0.1 % and the average absolute error is 3.0 %.

Table 1 Average errors per intersection

Intersection	Average error (s)	Average absolute error (s)
101	−0.7	6.2[a]
102	0.0	9.2
104	−0.6	11.4
Total	0.0	8.9

[a] When a correction is not applied for the neighboring intersection switching to a different control mode the average absolute error for intersection 101 goes up to 8.1 s

Note that 103 is just a pedestrian crossing with only public transport being allowed to use the street coming from the south. Therefore, it has no complete logging of the signal phases and only detection counts could be acquired. Apart from the missing data of intersection 103, some corrections had to be applied for certain external factors. Intersection 102 went to vehicle actuated mode at the 9th of March. Therefore, all data after the 9th was discarded. This also had a significant effect on the neighbor intersection 101, and the same correction was applied to this data set. Another issue was found with the default stance when there is no traffic on intersection 101. On the street it stayed in a stage with only signal group 8 and a long maximum green time, while in simulation it went to the stage with signal group 2, 8 and all parallel pedestrians and bicycles with a shorter maximum green time. Therefore, those signal groups were discarded from the data set.

Other inaccuracies of the model for which the data set was not corrected can be considered directions for future improvement:

- Pedestrians and bicycles were counted by the amount of times the push button was used. However, this is an inaccurate measure as a pedestrian may get impatient and push multiple times and a second pedestrian arriving may not push at all. The time between the start of red and the first time the button gets pushed would be a better indication for the arrival rate.
- Public transport priority and the respective bus schedules were not included in the simulation. These priority calls, even though not frequent can have a significant impact on the intersection.
- An extension loop at intersection 104 would sometimes freeze in the occupied state. Therefore, the corresponding signal group got a lot more green time than in the simulation, which had no broken loop. Conflicting signal groups on the other hand had fewer green. Correction for this would be difficult, but the system could create a warning that a detection inconsistency was detected.
- No special train functionality at intersection 104 was implemented in the simulation. When the railroad crossing to the south of 104 closes, special priority is given to cars coming from the south to ensure the railroad crossing to be free of cars. Additionally, after the railroad opens again, there is again a special priority for these vehicles as they have waited for a long time at the railroad already.

8 Conclusion

The paper has shown a method of coupling Imtech proprietary controllers to SUMO. The architecture used the same dll as other simulators use to couple to these controllers and therefore enables a user to freely select the preferred simulation software. For the interface, the challenges of different methods of assigning IDs to signal groups, signal heads and detectors between controllers and SUMO were overcome with a translation xml and a naming convention. On the SUMO side some extra variables were added to the TraCI interface to be able to access lane area

detectors as well. A test network that was implemented both in Vissim and SUMO showed that the results do not differ more than could be explained by different vehicle model configuration parameters.

An extension to combine this work with dynamic traffic demand patterns demonstrated the possibilities of creating an online micro-model. The results show that it is possible to model with a mean absolute error of 3.0 % for the resulting total red time per signal group in 5 min intervals. With this accurate online model, different traffic management strategies can be tested online before actually effectuating them. It also allows for testing multiple potential strategies in parallel to assist in the decision of which strategy to select.

Open issues identified during the work were problems with counting vehicles on the multi-entry multi-exit detectors and slow response to detector status requests through TraCI. Both issues will be taken up with the SUMO development team.

References

1. Bera S, Krishna Rao KV (2011) Estimation of origin-destination matrix from traffic counts: the state of the art. Europ Transp 49:3–23
2. Blokpoel RJ (2014) Network conversion for SUMO integration. In: 2nd SUMO conference, Berlin, Germany
3. Hoogendoorn SP, Bovy PHL (2001) State-of-the-art of vehicular traffic flow modelling. J Syst Control Eng 215:283–303 (1 June 2001)
4. Koenders E, in't Velt R (2011) Cooperative technology deployed. ITS Europe, Lyon, France
5. Krajzewicz D et al (2013) COLOMBO: investigating the potential of V2X for traffic management purposes assuming low penetration rates. In: ITS Europe congress, Dublin, Ireland, 4 June 2013
6. Liu H et al (2009) A neurak network model for travel time prediction. In: IEEE conference on intelligent computing and intelligent system
7. ns-3 (2014). ns-3 project web-pages.http://www.nsnam.org/. Accessed 11 Feb 2014
8. Seungkiri B, Hyunmyung K, Yongteak L, Gangwon L (2001) Multi-vehicle OD trip matrix estimation from traffic counts. J East Asia Soc Transp Stud 4(2)
9. Van Vliet K, Turksma S (2013) ImFlow: policy-based adaptive urban traffic control first field experience. ITS Europe, Dublin, Ireland
10. Xiao H et al (2005) Methodology for selecting microscopic simulators: comparative evaluation of AIMSUN and VISSIM. Research report, University of Minnesota