ITSUMO: an Agent-Based Simulator for ITS Applications

Ana L. C. Bazzan†, Maicon de Brito do Amarante†, Tiago Sommer and Alexander J. Benavides†
†Instituto de Informática, UFRGS
C.P. 15064, 91501-970, P.Alegre, RS, Brazil
{bazzan,mbamarante,ajbenavides}@inf.ufrgs.br

***Abstract*— One way to cope with the increasing demand in transportation networks is to integrate standard solutions with more intelligent measures. This problem has been approached from different sides such as the study of the assignment of the demand in the network, and the investigation of the effects of control measures. However, given that most of these approaches are complex and deal with different levels of abstraction of the original problem, there has been few attempts to address both demand and control in a single tool. This paper presents an overview of ITSUMO, a microscopic traffic simulator whose implementation uses agent technologies with a bottom-up philosophy in mind. We give an overview of the system and some details of its modules (simulation kernel, data management, control, driver and routing, and visualization), followed by an example of its use.**

I. Introduction and Motivation

The second half of the last century has seen the beginning of the phenomenon of traffic congestion. This arose due to the fact that the demand for mobility in our society has increased constantly. Traffic congestion is a phenomenon caused by too many vehicles trying to use the same infrastructure at the same time. The consequences are well-known: delays, air pollution, decrease in speed, unsatisfaction which may lead to risk manouvers thus reducing safety for pedestrians as well as for other drivers.

The increase in transportation demand can be met by providing additional capacity. However, this might no longer be economically or socially attainable or feasible. Thus, the emphasis has shifted to improving the existing infrastructure without increasing the overall nominal capacity, by means of a better utilization of the available capacity. Two, complementary, measures can be taken. In traffic engineering terminology these are associated with management of the *demand* (users, drivers) and *supply* (infrastructure, control). The set of all these measures is framed as Intelligent Transportation Systems (ITS).

In the last years there has been some proposals for simulation platforms that are flexible enough to test ITS techniques and approaches. Some (e.g. Paramics, AISUM, VISIM, EMME2, Dracula) are based on classical models of simulation and are commercial tools. With the appearance of a new simulation paradigm – agent-based simulation – it is now possible that traffic experts and other users develop their own applications. This has been achieved at some extent (e.g. [7], [13], [16], [1], [5], [6], [15], [17]) but these tools are goal-directed meaning that they were built for (more or less) specific purposes. One of the notable exceptions is MATSim (`www.matsim.org`). However, MATSim's simulation paradigm is queue-based, traffic signals are very simple, and drivers are not fully autonomous (e.g. during replanning).

Most of previously mentioned works have one or more of the following drawbacks: they are not fully agent-based; they rely on strong simplifying assumptions, they do not consider both control and assignment of demand as a whole process (except in [17] but here the integration only refers to their specific market-based approach).

Therefore there is a lack of support for traffic experts who want to implement and test their own solutions (e.g. artificial intelligence (AI) based approaches for optimization or broadcast of recommendation). These experts can neither extend commercial tools (except for some API-based modules that i) are not totally flexible and ii) represent an additional purchase cost) nor use the available free tools as these deal only with pieces of the whole problem.

This way there is still the need of an integrated platform that: be fully based on the autonomous agent paradigm of simulation; be open-source and user-friendly; consider the effects of both control measures on driver's reasoning and vice-versa.

The present paper describes ITSUMO (Intelligent Transportation System for Urban Mobility), an open-source tool that addresses these issues. It allows the modeling of traffic actors (drivers, traffic lights, and even autonomous vehicles) as autonomous agents; it deals with short term control of traffic lights and with *en route* replanning by drivers; thus it permits the study of co-effects of both demand and supply. This is achieved by means of AI techniques in general and of agent-based techniques in particular.

With the increase dissemination and computing power of mobile devices, it is now possible to execute distribued AI applications for various situations: intelligent routing using algorithms that do not rely on full-knowledge; planning under constraints and restricted communication and information; distributed optimization of traffic lights. For instance, it is possible to define drivers as intelligent agents and to plug each driver model. This approach is in contrast with current models, which are purely reactive and ignore drivers' mental states (informational and motivational data). Also, it is possible to plug reinforcement learning based control for traffic lights.

An earlier version of ITSUMO was presented as a demo in the AAMAS conference [14]. However, although ITSUMO has also been used to investigate route choice scenarios, e.g. in [3], the focus has been primarily on control. The current version was extended in the sense that it now allows

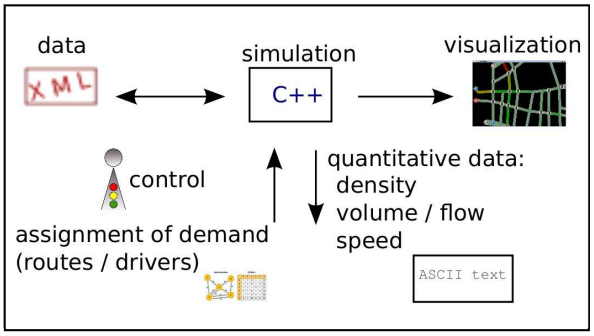

Fig. 1. High-Level View of the Simulation Modules

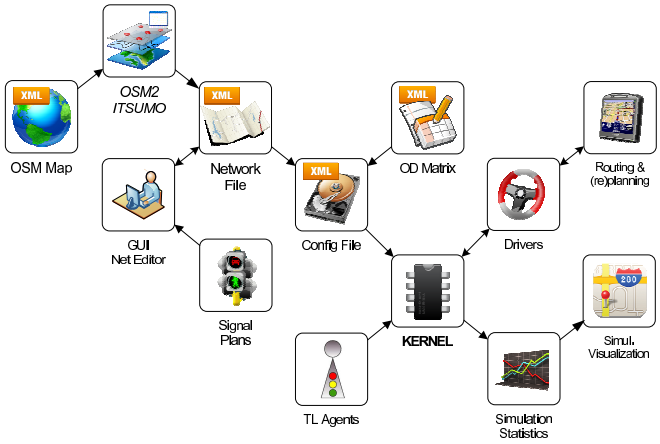

Fig. 2. Specific Functionalities of the ITSUMO Framework

modeling of both control measures and drivers reaction to them, as well as routing techniques. Moreover this is provided as non-comercial code and is one of the few tools that are truly agent-based (thus microscopic). As shown in the next section, the simulation kernel is responsible for handling the movement of vehicles. Other modules support the agent-based modelling of demand and supply.

In the next section details of the simulator and an overview of the main modules are given. Section III briefly revisits the main aspects by means of a case study where traffic lights adapt and where autonomous drivers can plan their routes. The last section concludes the paper.

II. Description of the Simulator

ITSUMO is composed by five modules: database, the simulation kernel, control, demand (assignment and drivers' definition), and the output module (visualization and statistics). Figure 1 shows how these modules interact.

In order to run a simulation, the topology must be stored as an XML file. After running the simulation, two optional outputs can be used: either through on-screen visualization (macroscopic or microscopic) or via dump of various data files. Other, optional, modules are the insertion and control via signal plans, and the assignment.

Figure 2 provides more details about specific funcionalities that will be discussed in the next sections. As discussed below, ITSUMO allows data configuration in various ways, and also provides the basic interfaces for its extension. For example, it is possible to extend the framework adding new routing algorithms or new traffic light control methods. Currently these alternatives are as in Table I. We discuss them in the next subsections where we give details about the modules. Before, we remark that, because the simulator is extendable, the user may add its own code for both communication (e.g. between controllers and drivers), and for definition of the infrastructure (beyond the one already provided).

A. Microscopic Simulation Model and Simulation Kernel

In contrast to macroscopic models of traffic simulation (which are mainly concerned with the movement of platoons of vehicles, focusing on the aggregate level), in the agent-based paradigm each object can be described as detailed as desired, thus permitting a more realistic modeling of drivers' behavior for instance. In the agent-based approach both travel and/or route choices may be considered, which is a key issue in simulating traffic since those choices are becoming increasingly more complex. Also, individual traffic lights can be modelled according to several approaches, from classical off-line coordination to recently proposed ones (negotiation, communication-free, via game theory, reinforcement learning, swarm intelligence, etc).

In order to achieve the necessary simplicity and performance, ITSUMO uses the Nagel–Schreckenberg cellular–automata (CA) model [9] for traffic movement (aka. Na-Sch model). In short, each road is divided in cells with a fixed length. This allows the representation of a road as an array where vehicles occupy discrete positions. The movement follows rules that represent a special form of car–following behavior. This simple, yet valid microscopic traffic model, can be implemented in such an efficient way that is good enough for real-time simulation and control of traffic.

Hence the kernel of the simulator (implemented in C++) is based on the CA model. The simulation occurs in discrete steps and is implemented as a series of updates in the vehicles' positions in the network. Each update in a node

TABLE I
Alternatives for Topology and Signals Edition, Control, Routing Algorithms, Conditions for Routing and Planning

topology	manual		
	OSM		
traffic light creation	manual		
	auto		
traffic light control	fix time		
	greedy		
	RL		
routing algorithms	Dijkstra		
	ARA*		
	Full dynamic batch change		
routing, planning and replanning	pre-trip	OD-based	
		Na-Sch + FC	
		Na-Sch	
	en-route	Na-Sch	
		congestion based	full knowledge
			partial knowledge

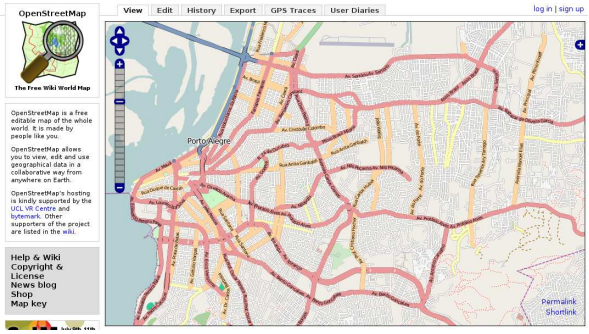

Fig. 3. OSM Export Feature (here for Porto Alegre, Brazil)

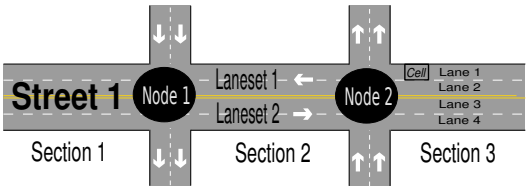

Fig. 4. ITSUMO Main Objects of the Topology

or traffic light may modify its current behavior.

B. Database Module

The information regarding the topology of the traffic network is stored in an XML file (see Figure 5). The database module creates, updates, and stores the static and the dynamic objects to be used in the simulation, both related to the infrastructure (supply) and to the demand. Regarding the former, the main attributes are: cartesian coordinates of intersections, streets characteristics (number of lanes, etc.); and signal plans (set of lane-to-laneset allowed movements). Regarding the demand, the dataset stores: insertion rate of vehicles at given nodes of the network; origin and destination of drivers, etc. Here we show the attributes related to the topology only; see Figure 4. For more details about these attributes please see [14].

As indicated at the top of Table I, this kind of data can be either entered manually – via a GUI, or be imported directly from the Open Street Map (OSM, `www.openstreetmap.org`) via the applicative "OSM2ITSUMO" (see Figure 2). Using the former method, each component in the network (e.g. as depicted in Figure 4 for a small example) is inserted by the user. This process is time-consuming and error-prone. It can however be used for small networks. For complex or big maps, the alternative is to use the latter method.

Because the XML format used in OSM is different from the one used in ITSUMO, we provide a parser to get the necessary data from OSM. The user just has to select the portion of the OSM map s/he wants to use (typically a bounding box as in Figure 3), export this to an XML (in OSM format), and run the parser. The output is then a new XML in the ITSUMO format (i.e. nodes, streets, sections, lanesets, and lanes are created with their corresponding attributes).

It is important to remark that most of the streets in OSM have a tag to classify them (motorway, primary, secondary,

```
<nodes>
 <node>
   <node_id> 2 </node_id>
   <node_name> n0 </node_name>
   <x_coord> 6250.0 </x_coord>
   <y_coord> 6000.0 </y_coord>
  </node>
 ...
<traffic_lights>
 <traffic_light>
   <traffic_light_id> 161 </traffic_light_id>
   <located_at_node> 14 </located_at_node>
   <signalplans>
  <signalplan>
 <signalplan_id> 1 </signalplan_id>
 <phases>
   <phase>
    <phase_id> 1 </phase_id>
    <iteration_start> 0 </iteration_start>
 ...
<section>
 <section_id> 74 </section_id>
 <section_name> (g2) &lt;-&gt; (n2) </section_name>
 <is_preferencial> false </is_preferencial>
 <delimiting_node> 25 </delimiting_node>
 <delimiting_node> 4 </delimiting_node>
 <lanesets>
   <laneset>
    <laneset_id> 75 </laneset_id>
    <laneset_position> 1 </laneset_position>
    <start_node> 25 </start_node>
    <end_node> 4 </end_node>
    <turning_probabilities>
    <direction>
   <destination_laneset> 78
   </destination_laneset>
   <probability> 100.0 </probability>
    </direction>
    </turning_probabilities>
   <lanes>
    <lane>
     <lane_id> 76 </lane_id>
     <lane_position> 1 </lane_position>
     <maximum_speed> 10 </maximum_speed>
     <deceleration_prob> 0.0
     </deceleration_prob>
    </lane>
   </lanes>
  </laneset>
 </lanesets>
</section>
 ...
```

Fig. 5. XML file (partially) describing a traffic network in ITSUMO

residential, etc.). Thus, our parser can import streets that match one or more of these tags. For instance it is possible to import just the main arterials present in an OSM map, or all links, or links in any other degree of abstraction. Hence, by importing OSM maps (in different abstraction levels), ITSUMO permits the use of real-world maps.

Similarly to the definition of the network topology, there are two ways to create signal plans in ITSUMO (notice however that the latter is optional; the simulation can be run without defining traffic lights and corresponding plans). As indicated in Table I, plans can be created manually and automatically. In the former, the user is requested to enter the full definition of all signal plans for each signalized intersection. Figure 6 depicts the GUI provided for creation/edtion. For each plan, phases must be defined (using the mouse to connect incoming lanes to outgoing lanesets), their cycle times, and splits (not shown). This is again time-consuming. If the network has too many signalized intersections, ITSUMO's automatic signal plan generator can

be used. The user has to inform the cycle time, which will be splitted equally among all defined movements. This of course generates only one, simplified, plan. However, users can edit the plan (using the same GUI used for manual edition) and modify it as desired.

The database also stores other objects such as sources, sinks, turning probabilities, etc. Due to lack of space we refer the reader to [14].

C. Control: Traffic Light Agent Module

In ITSUMO the control of traffic lights is implemented and executed via traffic light agents. These can control one or more intersections, using the same kind of control method or different ones. This is so because each agent is independent i.e. decides its own action taking into account the information available and using its own algorithm. Notice however that the information available may come from other traffic light agents so that it is not the case that each agent uses exclusively local information. In summary, traffic light agents may be heterogeneous and handle congestion in different ways.

Some basic classes for creating traffic light agents ("TL Agents" in Figure 2) have been implemented in order to facilitate the development of traffic controllers. These agents are organized in a data structure that is kept separated from the simulator. Thus, the user does not need to manipulate the kernel code. Moreover if any user wishes to code its own control method, this can be easily done.

A communication is established between the agents and the kernel using sockets. This permits the exchange of information about traffic status and control actions. The former can be e.g. number of stopped vehicles, density, speed etc. of the lanes under control by the given agent. The agent can then send a control action back to the simulation kernel (normally this control action is the number of the signal plan that the should be run at a given intersection).

These control actions can be implemented by the user as desired. So far we provide the following control methods: fixed time (i.e. only one signal plan is used and no change is made so that in fact no control is actually performed); greedy (gives priority – i.e. more green time – to the phase with more congested approaching lanes); and reinforcement learning based approaches (e.g. Q-learning). For more details about our previous uses of these approaches we refer the reader to [4] (greedy) and [2], [11], [12] (use of Q-learning). Other methods already tried were swarm-intelligence [10] and reinforcement learning with function approximation [18].

D. Demand Assignment and Driver Definition Module

1) Routing: Demands are normally represented by an OD (origin-destination) matrix that results from some survey or other kind of measurement of demand. ITSUMO can also generate synthetic demands (see "OD Matrix" in Figure 2). This can be done assigning either uniform probabilities to all nodes or to a set of selected nodes, or specific origin and destination probabilities to selected nodes. In the latter, node A for instance may originate 20% of the trips and collect 5% of them, while node B originates, say, 2% and collects 30%.

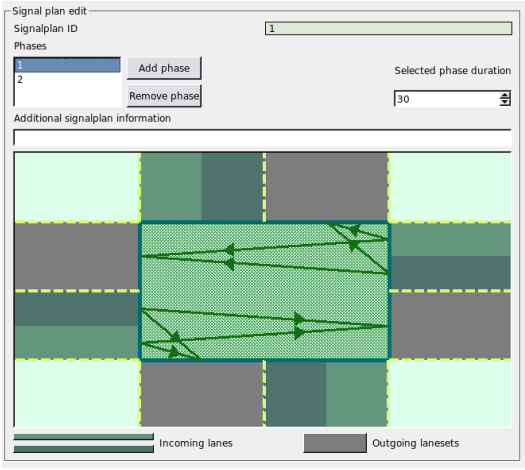

Fig. 6. GUI for editing signal plans

For each trip, a vehicle is generated and a route is assigned. This is in sharp contrast with the basic Na-Sch model where vehicles are treated as individual particles *without* a route. Rather, they are routed at each intersection with a probability to turn left, right, or continue in the same street.

So far ITSUMO has basically allowed the creation of vehicles as Na-Sch particles, or provided a GUI to define route for only a handful of routes, those that were assigned to the so-called "floating cars" (FC). This process of route definition was manual and could not be done for many vehicles.

With this new version, to generate routes for vehicles, ITSUMO can use various algorithms as shown in the third block of Table I. For example, ARA* stands for Anytime Repairing A* [8], a heuristic search algorithm; the third is a dynamic shortest path algorithm that uses dynamically changing quantities (e.g. traffic volume) as links'weights.

No matter the algorithm used, the routing can be done either in a centralized way (e.g. routes are computed in a centralized manner and are assigned to vehicles), or in a decentralized way. The centralized case is trivial and is performed as commonly used in commercial simulators: given an OD matrix, an algorithm computes routes for each driver, simulates the journeys, and performs further re-assignments until an equilibrium is found.

In the decentralized case, there is a de-coupling vehicle-driver because this allows the specification of several classes os drivers' behavior ("Drivers" in Figure 2). Here, the driver computes its own route based on a given strategy and on local knowledge. Therefore we refer to this as planning and discuss it in the next subsection.

2) Driver Definition: Modeling drivers' behavior can be approached in different ways, depending on the purpose of the simulation. In some cases, the objective is to simulate the collective or macroscopic behavior. However, this behavior emerges out of individual ones. Simple algorithms, like the CA model, can be used to describe the movement of vehicles. However, this model does not provide support for modeling

more sophisticated driver behaviors such as those based on route planning or en-route decision, which is appealing to AI practicioneers.

Next, the recents extensions that were made to ITSUMO in order to allow the definition of classes of drivers are discussed. This discussion concentrates here on two aspects: the pre-journey planning and the en-route (re-)planning.

The centralized assignment discussed in the previous section works only to the extent that the full rationality assumption is considered: drivers want to maximize their individual utilities. This economometric model considers neither bounded-rationality nor individual preferences. However it is a relatively strong assumption that drivers know the whole traffic network, not to speak about the traffic status of each links at each moment. The first assumption can only be accepted if all drivers are known to have GPS-based devices to guide them. The second (accurate, instantaneous knowledge of the status of each link) is still far from reality. Even when this is deployed, it is questionable whether drivers can indeed process all the information. Therefore the centralized routing only addresses macroscopic investigations e.g. those that are carried out for urban planning purposes. It is not efficient to model real behavior of individual drivers.

In the decentralized route computation, it is assumed that the driver itself will plan its journey, given its (partial) knowledge of both the traffic network and of the current traffic status.

ITSUMO has a series of methods to allow the implementation of routing at the driver level. So far, without need of further coding, it is is possible to use the algorithms that are mentioned in Table I, both in centralized and decentralized variants. Of course, in the latter case, links' weights may differ from driver to driver as they are local perceptions.

3) Drivers and En-Route Replanning: One of the features of an autonomous driver is its ability to replan during the trip when facing congestion. In classical centralized approaches, this is hard to do due to the fact that each driver may have its own replanning strategy, own knowledge about traffic status, as well as own preferences and idiosyncrasies. In order to facilitate these definitions, ITSUMO allows each class of driver to have its own profile.

In Table I (last block) we show the possibilities for *en-route* replanning. One possibility is the trivial Na-Sch re-routing of vehicles but here there is actually no planning because this vehicles are treated as particles that are randomly re-routed.

The actual replanning, the one that happens autonomously at driver's level is based on driver's perception of the congestion level. So far we assume that drivers only have local perception and hence partial knowledge of the traffic status. However, the simulator is prepared to deal with situations in which drivers have full knowledge.

En-route replanning can be done using one of the algorithms mentioned. In all cases, driver will compute a new route from the point where s/he starts to replan to the destination. If dynamic shortest path algorithms are used, then the current traffic status of the known links are used. For unknown links, the lenght is used instead.

Fig. 7. Macroscopic and Microscopic Visualization of a Simulation

This means that when a driver arrives at a link $e^i \in \mathcal{P}^j$, where $\mathcal{P}^j$ is the initially computed route of vehicle j, s/he evaluates how delayed s/he is when compared to the expected time. If the current time step is τ times higher than the expected time step, than the driver replans the route. Besides, the exact portion δ of the route where the driver considers replanning is also configurable. More anxious drivers will start replanning sooner. Therefore δ and τ (among other factors) define different classes of drivers. At the end of the simulation, one can evaluate the performance of each class of driver and compare these.

E. Ouptut Module: Statistics and Visualization

Sensors and detectors are used to collect information that is displayed during the simulation. Thus, sensors collect all sorts of information about the scenario being simulated, such as the lane occupation rate, the average vehicle speed in a street, in/out flow of vehicles in a specific laneset, etc.

The simulation output can be formatted according to the user needs. The most usual formats are the "cell map" and the "laneset occupation map". The former indicates which portions of the lane are occupied by which vehicle, providing the most detailed output possible. On the other hand, the "laneset occupation map" is a high-level output which specifies the rate of occupation (density) for each laneset in the network.

Users can visualize the simulation either in a macroscopic or in microscopic level (individual vehicles). Both can be seen in Figure 7 where the small figure at top left shows the microscopic visualization of one intersection. At a macroscopic level, the visualization considers only data that reflects the overall behavior of the network, providing an useful tool to capture the big picture of what is happening in a specific scenario. The microscopic level provides an interface through which one can see individual vehicles movement. A third kind of visualization is a plot of a subset of the vehicles route over a map using Google Maps.

TABLE II
TRAVEL TIME TO DESTINATION: COMPARISON

Nb. of Vehicles	Time for all vehicles to reach their destination		
	No light	Fixed time	Greedy
2000	600	3100	1500
2500	900	3000	1600
3000	>3600	4100	2000
3500	>3600	5200	3100

III. CASE-STUDY: AGGREGATING INTELLIGENCE TO TRAFFIC SIMULATION

In order to illustrate the use of ITSUMO with new facilities for demand handling, we discuss a case study. This deals with the city of Porto Alegre (30°05'S, 51°10'W) in Brazil (Figure 3). The downtown part of the city was selected and exported from OSM and parsed to ITSUMO format using "OSM2ITSUMO". We discuss scenarios with and without traffic lights. When these are present, the signal plans were generated automatically using cycle length of 60 seconds, with uniform green time for all phases. Thus if an intersection has only two phases, each receives 30 seconds of green time. The traffic light agent runs a greedy strategy that increases this split for the most congested approach (and decreases the same proportion for the other approach).

Overall, that area of the city comprises 159 nodes (96 having traffic lights), 225 links totalizing 39K meters, and we have varied the number of vehicles from 2K to 3.5K. We remark that since each cell has 5 meters, the network holds up to approximately 8K vehicles. For the purpose of illustration we have simplily assumed that the demand is uniformly distributed among all nodes. Other simulation parameters are: unless mentioned, the number of simulation steps is 3600; the cell size is 5 meters; and the maximum speed is 3 cells per simulation step (roughly 54 km/h).

We discuss next a simple example where drivers plan their routes using the Dijkstra algorithm taking the lenght of the link as weight. Although the simulator allows the output of the load of all links in order to generate histograms of links usage, here we just show the time necessary for all vehicles to arrive at their destinations, for the situation without traffic lights, with traffic lighs and fixed time plans, and with greedy control (Table II). For the case with traffic lights, due to delays, the trips take longer and hence the simulation time was raised to 6000 time steps.

One sees that with a occupancy of roughly 50% (which is very high given that it refers to the whole network) many drivers do not arrive to their destinations within the simulation horizon. When traffic lights are included, all drivers arrive even if, for low volume, trips take longer on average. This happens because the signal timings are fixed (uniform split for all approaches) causing unnecessary delays. When greedy adaptive signal plans are considered, the situation improves significantly as the trips take less time and all drivers quit the simulation before its end.

This simple, yet valid scenario, was given just for the sake of illustration of the use of the simulator, and to show what kind of results one may extract. It is not the purpose here to make comparisons with other tools, even because most of them do not allow, for instance, en-route replanning.

To give an idea of scalability, we remark that we have also run simulations that cover the whole city of Porto Alegre. In this case we did not include all intersections and all streets, just the main arterials. These arterials however can hold around 100K drivers.

IV. CONCLUSION

This paper has presented ITSUMO, an open source microscopic traffic simulator that allows modeling of individual drivers or classes of drivers along with the implementation of different signal control strategies.

We plan to extend ITSUMO to consider other kinds of information such as those related V2V communication and information providing via internet and/or mobile phone in order to compare the performance of informed versus non-informed drivers.

ACKNOWLEDGMENTS

This project and the authors are funded by CNPq. We thank the referees for suggestions and corrections.

REFERENCES

[1] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. W. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of MATSim-T. In *2nd TRB Conference on Innovations in Travel Modeling, Portland, June 2008*, 2008.

[2] Ana L. C. Bazzan, Denise de Oliveira, and Bruno C. da Silva. Learning in groups of traffic signals. *Eng. Applications of Art. Intelligence*, 23:560–568, 2010.

[3] Ana L. C. Bazzan and Franziska Klügl. Case studies on the Braess paradox: simulating route recommendation and learning in abstract and microscopic models. *Transportation Research C*, 13(4):299—319, August 2005.

[4] Ana L. C. Bazzan, Kai Nagel, and Franziska Klügl. Integrating MATSim and ITSUMO for daily replanning under congestion. In *Proceedings of the 35th Latin-American Informatics Conference, CLEI*, Pelotas, Brazil, September 2009.

[5] Ana L. C. Bazzan, J. Wahle, and F. Klügl. Agents in traffic modelling - from reactive to social behavior. In *Advances in Artificial Intelligence*, number 1701 in Lecture Notes in Artificial Intelligence, pages 303–306, Berlin/Heidelberg, 1999. Springer. Extended version appeared in Proc. of the U.K. Special Interest Group on Multi-Agent Systems (UKMAS), Bristol, UK.

[6] B. Burmeister, J. Doormann, and G. Matylis. Agent-oriented traffic simulation. *Transactions of the Society for Computer Simulation*, 14(2):79–86, 1997.

[7] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In N.R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, New York, USA, July 2004. New York, IEEE Computer Society.

[8] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime search in dynamic graphs. *Artif. Intell.*, 172(14):1613–1643, 2008.

[9] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221, 1992.

[10] Denise de Oliveira and Ana L. C. Bazzan. Swarm intelligence applied to traffic lights group formation. In Inês de Castro Dutra, Ricardo Choren, and Cláudia Maria Garcia Medeiros de Oliveira, editors, *Anais do VI Encontro Nacional de Inteligência Artificial (ENIA)*, pages 1003–1112. SBC, July 2007.

[11] Denise de Oliveira and Ana L. C. Bazzan. Multiagent learning on traffic lights control: effects of using shared information. In Ana L. C. Bazzan and Franziska Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 307–321. IGI Global, Hershey, PA, 2009.

[12] Denise de Oliveira, Ana L. C. Bazzan, Bruno C. da Silva, E. W. Basso, L. Nunes, R. J. F. Rossetti, E. C. Oliveira, R. Silva, and L. C. Lamb. Reinforcement learning based control of traffic lights in non-stationary environments: a case study in a microscopic simulator. In Barbara Dunin-Keplicz, Andrea Omicini, and Julian Padget, editors, *Proceedings of the 4th European Workshop on Multi-Agent Systems, (EUMAS06)*, pages 31–42, December 2006.

[13] Rosaldo Rossetti and Ronghui Liu. A dynamic network simulation model based on multi-agent systems. In F. Klügl, A. L. C. Bazzan, and S. Ossowski, editors, *Applications of Agent Technology in Traffic and Transportation*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 181–192. Birkhäuser, Basel, 2005.

[14] Bruno Castro da Silva, Robert Junges, Denise Oliveira, and Ana L. C. Bazzan. ITSUMO: an intelligent transportation system for urban mobility. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 1471–1472. ACM Press, May 2006.

[15] Kagan Tumer, Zachary T. Welch, and Adrian Agogino. Aligning social welfare and agent preferences to alleviate traffic congestion. In Lin Padgham, David Parkes, J. Müller, and Simon Parsons, editors, *Proceedings of the 7th Int. Conference on Autonomous Agents and Multiagent Systems*, pages 655–662, Estoril, May 2008. IFAAMAS.

[16] R.T. van Katwijk, P. van Koningsbruggen, B. De Schutter, and J. Hellendoorn. A test bed for multi-agent control systems in road traffic management. In F. Klügl, A. L. C. Bazzan, and S. Ossowski, editors, *Applications of Agent Technology in Traffic and Transportation*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 113–131. Birkhäuser, Basel, 2005.

[17] Matteo Vasirani and Sascha Ossowski. Exploring the potential of multiagent learning for autonomous intersection control. In A. L. C. Bazzan and F. Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 280–290. IGI Global, Hershey, PA, April 2009.

[18] S. J. Waskow and Ana L. C. Bazzan. Improving space representation in multiagent learning via tile coding. In A. C. R. Costa, R. M. Vicari, and F. Tonidandel, editors, *Advances in Artificial Intelligence – SBIA 2010*, Lecture Notes in Artificial Intelligence. Springer, 2010 (to appear).