Extending traffic simulation based on cellular automata: from particles to autonomous agents

Ana L. C. Bazzan, Maicon de B. do Amarante, Guilherme G. Azzi, Alexander J. Benavides, Luciana S. Buriol,
Leonardo Moura, Marcus P. Ritt, Tiago Sommer
Instituto de Informática, UFRGS
C.P. 15064, 91501-970, P.Alegre, RS, Brazil
Email: {bazzan,mbamarante,ggazzi,ajbenavides,buriol,lmoura,mpritt,tsommer}@inf.ufrgs.br

ABSTRACT

Cellular automata models for modeling of traffic movement assume that vehicles are particles without route. However, if one is interested in analysing microscopic properties, it is necessary to assign a route to each trip. This paper discusses the latest developments in the ITSUMO traffic simulator, which aim at modeling more sophisticated driver behaviors such as en-route decision-making. This was tested in two scenarios, one being a real-world traffic network. We make an extensive discussion about the effects of the use of various routing algorithms as well as demand and capacity, control measures, network topologies, and re-planning strategies.

1 Introduction

The traditional cellular automata (CA) model for microscopic modeling of traffic movement introduced by Nagel and Schreckenberg (1992), as well as its extensions, consider that the traffic demand (vehicles) are particles without route. Rather, vehicles are routed at each intersection with a probability to turn left, right, or continue in the same direction. For the purpose of generating a macroscopic picture of the traffic situation, this is a fair assumption. However, in reality, for each vehicle and trip, a route must be assigned if one is interested in analysing *microscopic* properties. For instance, the CA does not provide support for modeling more sophisticated driver behaviors such as route planning or en-route decision, which is appealing to AI practitioners.

In the present paper we aim at discussing the effects of the introduction of routing mechanisms in the CA model of traffic simulation that underlies the ITSUMO (Intelligent Transportation System for Urban Mobility) simulator (Silva et al. (2006); Bazzan et al. (2010)). One motivation behind ITSUMO is to allow the use of AI and autonomous agents techniques to address the increasing complexity of problems related to urban mobility. For instance, one may simply plug a model for a class of drivers. 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. Hence, ITSUMO deals with short term control of traffic lights and with *en route* re-planning by drivers; thus it permits the study of co-effects of both demand and supply in a more natural way. Since the control modules were already discussed in previous papers, here we focus on the discussion about the latest additions, which are related to the demand and routing mechanisms and algorithms.

This paper is organized as follows. In the next section we discuss related works. Given that there is an extensive list of references that could be quoted here, thus making this article too long, we focus on some background ideas and on microscopic simulators that are agent-based. Section 3 gives an overview of the ITSUMO simulator and focus on the recent extensions. To illustrate our approach we use two scenarios that are described in section 4. These are then presented and discussed in Section 5. We give some concluding remarks and outline the future work in Section 6.

2 Related Work

In the last years there has been some proposals for simulation platforms that are flexible enough to test ITS (Intelligent Transportation Systems) approaches. Some (e.g. Paramics, AISUM, VISIM, EMME2) 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. Dresner and Stone (2004); Rossetti and Liu (2005); van Katwijk et al. (2005); Balmer et al. (2008); Vasirani and Ossowski (2009)). However, 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 re-planning). Moreover, most of these works do not consider both control and assignment of demand as a whole process (except the latter but there the integration only refers to their specific market-based approach).

Our aim with ITSUMO is to fill this gap with a non-commercial code that is truly agent-based (thus microscopic). An earlier version of ITSUMO was described in Silva et al. (2006). The current version was extended Bazzan et al. (2010) to allow modeling of both control measures and drivers reaction to them.

3 Approach and Description of the Simulator

The approach we follow is completely agent-based. Actors in the urban environment (drivers, traffic lights, and autonomous vehicles) are modeled as agents. ITSUMO is composed by five modules: database, the simulation kernel, control, demand (assignment and drivers' definition), and the output module (visualization and statistics). Next, we briefly present the existing modules, focussing on the demand part.

3.1 Basic and Control Modules

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.

In order to achieve the necessary simplicity and performance, ITSUMO uses the Nagel–Schreckenberg cellular–automata (CA) model Nagel and Schreckenberg (1992) 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.

Database: The information regarding the topology of the traffic network is stored in an XML file. 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. Topological data (i.e. map attributes) can be either entered manually, or be imported directly from the Open Street Map (OSM, `www.openstreetmap.org`). The database also stores other objects such as sources, sinks, turning probabilities, etc. Due to lack of space we refer the reader to Silva et al. (2006).

Control: In ITSUMO the control of traffic lights is implemented and executed via traffic light agents. A communication is established between the agents and the kernel using sockets. This tells the kernel to run the action (signal plan) selected by the traffic light agent. These control actions can be implemented by the user as desired. Several were already tested by us, especially based on reinforcement learning, as in Bazzan et al. (2009).

Output: Sensors and detectors are used to collect information that is displayed during the simulation, such as the lane occupation rate, the average vehicle speed in a street, travel time, etc.

Users can visualize the simulation either in a macroscopic or in microscopic level (individual vehicles).

3.2 Demand Assignment and Routing

Demands are normally represented by an OD (origin-destination) matrix that results from some survey or other kind of measurement of demand. 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. This means that particles are not actually autonomous agents since they do not pick their own routes.

ITSUMO allows the following forms of demand handling: the creation of vehicles as Na-Sch particles; manual definition of a handful of routes; automatic generation of routes using various algorithms such as Dijkstra, A*, ARA*, anytime and dynamic shortest path algorithms. Next we give a brief overview on the less known of these algorithms for shortest route computation.

Both Dijkstra and A* algorithms are satisfactory in static problems. This is not the case when routes must be computed for every driver taking into account the actual cost of each link in the network. Given that these links depict frequent changes in such costs, efficient algorithms must be used. Moreover, in very large networks, there is no time to find the best route. In this case, anytime algorithms are helpful because a route can be computed, which is the best possible solution given a time bound. Anytime algorithms also make sense because it is usually the case that the costs associated with each link will change making expensive computations quickly out-of-date. In such cases it is interesting to compute a partial route that is both fast and inexpensive.

Likhachev (2005) presents three variations of the A* algorithm: an anytime variation, a dynamic one, and a variation that is both dynamic and anytime. Anytime Repairing A* (ARA*) is the anytime variation. It introduces a weight to control the lower bounds of A* and produces a solution with a controlled sub-optimality bound. It finds a sub-optimal solution quickly with a loose weighted bound in the first search. Later, when more time is available, it tightens the bound and reuses previous search efforts until it produces an optimal solution. Lifelong Planning A* (LPA*) is the dynamic variation. The initial LPA* search is the same than A* search. When there are changes in link costs, LPA* updates them and executes the search again. Subsequent LPA* searches reuse previous valid search efforts to find an updated optimal solution. Anytime Dynamic A* or Anytime D* (AD*) is both anytime and dynamic. It combines the variable weighted lower bound (ARA* property) and the update of changed costs (LPA* property). The initial solution can be improved after the trip of any driver has started, because there were changes in links' costs, or because the initial solution was not the best possible.

No matter the algorithm used, routing can be done either in a centralized way (e.g. routes are computed by a central entity and are assigned to vehicles), or in a decentralized way. The *centralized* case is trivial and is performed as 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, the driver computes its own route based on a given strategy and on local knowledge. Therefore we refer to this as *local* planning.

Besides, routes can be computed either in a static or in a dynamic way meaning that either the length of a link is used as cost, or this cost is computed based on the current state (occupation) of the link. For the latter, a cost function was devised, which considers occupation of the links as a kind of inflated length. This is based on the maximum possible speed given a particular occupation of the link. The maximum speed V a vehicle may reach in a link i is given by $V = min\{v_{max}, (N_c^i - N^i)/N^i\}$, where v_{max} is a parameter of the Na-Sch CA model (basically it is the maximum permitted speed of the vehicle, which can only be achieved under free flow); N^i is the current number of vehicles in the link; and N_c^i is the number of cells in the link (considering all lanes), i.e. the maximum number of vehicles that fit in it. The inflated length L^i of link i is then computed as $L^i = l^i \times \frac{v_{max}}{V}$ where l^i is the length of i.

To illustrate the idea, let us imagine that $N^i = 100$ and $v_{max} = 3$. With less than 25 vehicles in i, all of them may travel at v_{max}, hence the link weight is $L^i = l^i = 100$ (no penalty). Otherwise the link is penalized exponentially (e.g. for $N^i = 90$ and hence $L^i = 100 \times 3/0.11 = 2700$).

3.3 Drivers and En-Route Re-planning

One of the features of an autonomous driver is its ability to re-plan during the trip when facing congestion. En-route re-planning can be executed using one of the algorithms mentioned in Section 3.2. In all cases, a driver will compute a new route from the point where s/he starts to re-plan, up to the destination. Henceforth when we refer to re-planning we mean *en-route* planning. In this case, the current traffic status of the known links are used. For unknown links, the length is used instead.

So far we have implemented two strategies for triggering re-planning. One is called *intersection re-planning* (IR) while the second is *delay re-planning* (DR). IR means that drivers may re-plan at every intersection. DR is based on a driver's current delay. In this case, 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 re-plans the route. In order to use DR, the driver uses its perception. As mentioned before, we assume that a reasonable perception is the one that the driver can have locally. Thus, drivers' perception was limited to two links ahead (from its current location). For links farther than this, links' costs are assumed to be their respective lengths.

3.4 Remarks about the Approach

In this section we have outlined the main features of ITSUMO, now extended to include various routing algorithms to tackle the demand part. We remark that both the control and the demand models are fully agent-based. Although the user may or may not use the totality of the data and knowledge (thus simulating a centralized approach), it is also possible to let driver agents (or, for the sake of control, traffic signal agents) have access to only local knowledge (thus simulating a distributed process where agents just have local data gathered by means of sensors). Of course in the case of drivers, it is a reasonable assumption that nowadays the full map of the traffic network is easily accessible (e.g. if we assume that GPS devices are wide-spread).

In order to illustrate the effects of the various design possibilities, the next section discusses two scenarios and the respective results. Both scenarios are thought as typical commuting scenarios where drivers select a route from an origin to a destination. Both depart from the simple binary choice scenario frequently seen in the literature (e.g. Klügl and Bazzan (2004)). Therefore they deal with route choice in a network with a high set of possible routes, as it is the case in real-world scenarios.

The first scenario is a nearly regular 6x6 grid.In this grid all links have the same capacity, except for those belonging to the main avenue where there is a higher capacity. This however does not change significantly the regular characteristic of the grid. We decided to use this grid in order to compare the results with previous papers using this scenario, in which we have not used ITSUMO in its full.

For instance, in Bazzan et al. (2009) this scenario was used to illustrate the integration of ITSUMO and MATSim. In this case, the demand part was handled by MATSim, using a queue-based simulation model. Therefore it was not completely microscopic. A virtual queue was used where the actual position of the vehicle in the link does not matter. In opposition to this, here use use cellular automata. Another difference between MATSim and the current version of ITSUMO is that in ITSUMO the re-planning is completely at local level i.e. the driver itself decides to re-plan or not. MATSim used a scheme that simulates a centralized mechanism determining that a given percentage of the drivers re-plan and select those which will do so. This is done as such because the learning mechanisms implemented in MATSim (e.g. genetic algorithms) need full knowledge that is not necessarily known to the drivers.

The second scenario is closer to real-world urban networks. It is taken from the city of Porto Alegre (POA) in Brazil, where we use the main arterials and avenues. By

using this second scenario we are able to show that real world networks are different from regular grids.

4 Scenarios

4.1 Grid 6×6

In the 6x6 grid all 60 links that are associated with the 36 nodes are one-way and drivers can turn in each crossing. Although it is apparently simple, this kind of scenario is realistic and, from the point of view of route choice and equilibrium computation, it is also a very complex one as the number of possible routes between two locations is high.

Due to the fact that each cell measures 5 meters and each link is 300m long, the grid supports nominally 4200 vehicles. Most links have a single lane, that is, may contain 60 vehicles. Five links, however, have three lanes and a capacity of 180 vehicles.

For every driver agent, its origin and destination are either randomly selected or based on an existing (non-uniform) OD matrix. In the former case, we call this an uniform demand. For the 6x6 grid, an uniform demand is created by assigning each node being origin with probability of 1/36; and similarly regarding destinations.

Regarding non-uniform demands, in this paper we use the following (already used by us in previous papers as mentioned). On average, 60% of the drivers have the same destination. Other links have, each, 1.7% probability of being a destination. Origins are nearly equally distributed in the grid, with three exceptions (three "main residential areas"). The remaining links have each a probability of 1.5%.

Besides these two types of demand, we have also performed simulations using the Na-Sch driver. These attempts to emulate uniform demand: there are sources on every node, all producing vehicle with the same probability. Sinks are also located on each of the 36 nodes, and remove vehicles with a probability of 1/36.

No matter the kind of demand used, the actual trips after the routes are determined, are combined with a simple kind of control, namely by means of traffic lights running signal plans with fixed time, or greedy strategies. Each node runs a signal plan, with a cycle length of 60 seconds and a split of 50% of green time for each traffic direction. The actions of the traffic light agents are: to run the default signal (in the fixed mode), or to modify the base plan in a greedy way allowing more green time for the more congested approaching lanes.

4.2 Real-World Network

Although the 6x6 grid is a realistic scenario from the point of view of routing (as the number of possible routes is large), it is a small scenario given that nominally 4200 vehicles can occupy the network. Larger scenarios were already tested in ITSUMO as in Bazzan et al. (2010), where the downtown part of the urban network of POA

Figure 1: 15 Main Origins and Destinations in the POA network

was used. In that case the network accommodates 8000 vehicles. In the present paper we aimed at extending these figures considerably thus we are using an extended portion of the same city, depicted in Fig. 1. We have opted to have only the main network of arterials. We decided to consider only arterials because we want to focus on the effect of routing and re-planning. Therefore we need to consider a large portion of the city (otherwise routing makes little sense). The more links used, the more drivers that have to be simulated, or we risk having low occupancy. This of course has an impact in the simulation time. Thus we opt to have less links (but the busy ones). In any case, the network already considers a high number of nodes and links.

As in the 6x6 grid, for this second network we discuss cases with and without the use of 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.

Overall, the network comprises 61 nodes (46 having traffic lights), 38 sections totalizing 76K meters, and we have varied the number of vehicles as much as possible. We remark that one section has several lanes (typically 3 in each direction) and since each cell has 5 meters, the network holds up to approximately 100K vehicles.

Similarly to the grid 6x6, for every driver agent, its origin and destination are either randomly selected or based on a non-uniform OD matrix. The uniform demand was generated by assigning the probability of $1/61 = 1.64\%$ to every node (both for origin and destination). Regarding non-uniform demands, the origins and destinations are concentrated in 15 main nodes that are depicted in Fig. 1. Due to lack of space we do not show the OD matrix but note that for instance almost 10% of the trips originate in a given node. This is in sharp contrast with the 1.64% in the uniform demand.

5 Results

In this section we present the main results regarding both networks, and discuss them (from Section 5.2 on) regarding effects of different types of demand, use of traffic lights, routing algorithm, etc. We start with the 6x6 grid,

discussing the baseline case i.e. the basic Na-Sch model, where vehicles are treated as individual particles without a route. For our purposes in this paper, this has little usage because these particles cannot be routed. Therefore we just show briefly what happens if we do allow only this kind of vehicles to populate the network (Section 5.1). After we discuss the effects of algorithms and control strategies in both networks, as well as perform a comparison between them.

5.1 Baseline: Drivers as Particles

The main metric that is used in both the 6x6 and the bigger network is the average travel time of all routed vehicles. Unfortunately this metric is not adequate in the case of Na-Sch particles because these have no route and, at each junction, there is a probability that an existing particle is consumed by a sink. Thus, for the Na-Sch particles we use a different metric, namely the time necessary for the last driver to quit the simulation. This gives us a rough idea of how long it takes for a given number of drivers to travel before they are all removed. Na-Sch particles are generated, at each of the 36 nodes for a fixed time period, with a given insertion rate. For instance, if it is 0.2, one can expect the insertion of about 1000 vehicles into the network if we let the source active for $1000/(0.2 \times 36) \approx 139$ time steps.

With this rate, it takes 1030 time steps for the last of the 1000 vehicles to quit the simulation. This time then increases slightly up to around 1400 time steps when the rate is 1.0.

Contrarily to this situation, as discussed in the next sections, travel time for drivers with routes increase much more abruptly. This means that particles have a stable behavior and in fact this behavior is much more related to how the sink removes them than to how they select a route. This is of course not realistic as it is not what is observed in the real-world.

5.2 Simulating Autonomous Agents

One first remark that has to be done relates to particular characteristics of the Na-Sch model used for the movement of vehicles. Because a gap must be considered between the vehicles, one generally cannot achieve more than roughly 50% occupancy of the network (meaning that for each vehicle there is one empty cell ahead). This is especially the case when no traffic lights are used. When they are present, stopped vehicles of course may eventually fill all possible cells in a link. This has an effect on the maximum number of vehicles in the network at any given time, as shown below. In general, given the nominal capacities of the 6x6 and POA networks (4200 and 100K), we can expect to fit around 2K and 50K vehicles respectively.

The second remark relates to the fact that we have not implemented any deadlock solver because this would imply ad-hoc and arbitrary decisions (such as "jumping" vehicles out of a link when in a deadlock situation). Therefore, when we start increasing the number of vehicles in the simulation, at some point deadlock situations may happen as vehicles block each others and none is able to move further. This is especially the case in the 6x6 grid which has only one lane in most of the links. In the POA network most of the links have 3 or more lanes but deadlocks occur there as well because in some parts the level of congestion is severe.

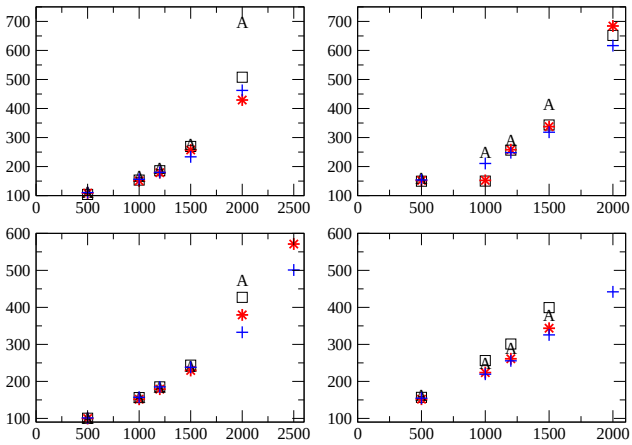

Figure 2: Average travel times in grid 6x6, for different number of drivers. Top left: uniform demand, no lights; Top right: uniform demand, with lights; Bottom left: non-uniform demand, no lights; Bottom right: non-uniform demand, with lights. □: Dijkstra; ∗: A*; A: ARA*; +: re-planning

Therefore, in the experiments discussed next, we start increasing the number of vehicles up to the point when deadlocks are noticed. In some cases the transition to deadlock regimes is nearly chaotic as for instance when 2K drivers are simulated in the 6x6 network, under uniform demand.

We now turn to the results using the routing mechanisms. The experiments were run in Intel(R) Core(TM) i7 CPU 860 / 2.80GHz with 8 GB of RAM. They run from 0.06 seconds (500 drivers, network 6x6) to 30 hours (50K, POA). As mentioned, the main metric here is the average travel time over all drivers that are created in the simulation. Travel times are given in simulation steps. In the cellular automata one time step is the time necessary for moving from one cell to another. For instance if a vehicle has speed 3, this means it moves 3 cells per time step. Given that the cell has 5 meters, this means 15 m/step. Thus, a travel time of, say, 1000 means a traveled distance of 15 km (with speed 3) or 5 km (with speed 1).

In the experiments, every driver may decide to re-plan. Here we just report results yielded by the IR strategy (see Section 3.3).

Figures 2 and 3 show travel times for various cases, for both networks. In both, the two plots at the top refer

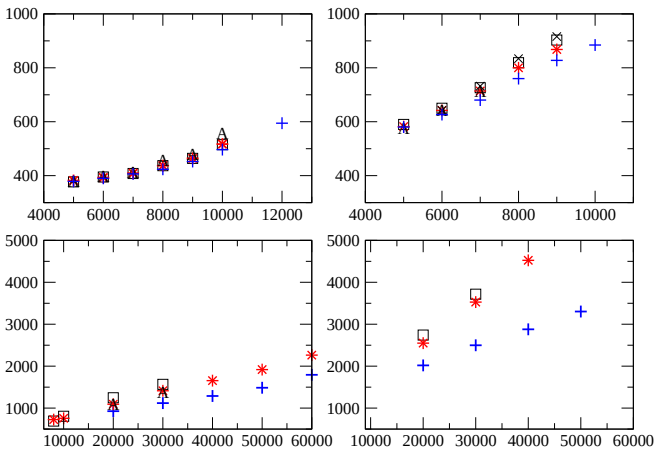

Figure 3: Average travel times in the POA network; Plots as in Fig. 2; □: Dijkstra; ∗: A*; A: ARA*; ×: greedy; +: re-planning

to demands uniformly distributed, whereas the two bottom plots refer to non-uniform OD matrices; left plots are with the activation of traffic lights, whereas the right ones refer to scenarios without traffic lights. Although we do not show error bars, we remark that the standard deviation of the averages we report here are mostly below 1%. We discuss the main effects of the changes of the various dimensions in the next sections.

5.3 Use of Different Routing Algorithms

As expected, algorithms belonging to the same family (e.g. Dijkstra and A*) yield equivalent travel times, with some remarkable exceptions. In general this can be explained by A* and Dijkstra selecting different kinds of routes in a regular grid. A* selects a more direct route trying to stick to a straight one. Dijkstra selects one among many routes that have the same optimal cost; usually this route is the same for different drivers. In particular, as seen in Fig. 2, left plots (no traffic lights), there is a significant difference in travel time between Dijkstra and A* for 2K drivers. This is explained by the fact that this is the transition to a deadlock situation in a regular grid. This transition is, as mentioned before, chaotic in this particular case probably due to the fact that there are in fact not many alternatives for drivers and hence deadlocks occur more frequently.

Apart from this, from figures 2 and 3, it is possible to see that the A* and Dijkstra yield approximately the same travel times. Also as expected, in general, the running times of A* are lower.

As for the ARA*, this was executed four times, tightening the weight from 2.5 to 1.0 each 0.5 to emulate a constraint regarding planning time. ARA* had a poor performance in network 6x6 because it selects the same routes for all the vehicles, similarly to Dijkstra, as mentioned. This is because there are many optimal routes with the same costs in a regular grid.

5.4 Uniform versus Non-Uniform Demand

Comparing top and bottom plots in Fig. 2, one notices that there is not a significant difference between the plots. This is so because nodes are regularly distributed. This has the consequence that the uniform demand generates traffic that is indeed nearly uniformly distributed. The OD matrix for this scenario, as mentioned before, foresees three major origins and one major destination but apart from this the rest is again uniformly distributed. This explains the relatively small difference between the two kinds of demands.

In the POA (non-regular) network (Fig. 3) however, many of the 61 nodes are concentrated in the center of the city. Having more nodes in this area makes it much more likely to originate and attract trips when the trips are uniformly distributed by node. This has a certain impact on travel times, but an even stronger impact in the appearance of deadlocks. In fact, looking at Fig. 3 one see that it was not possible to simulated more than 12K for the case of uniform demand due to existence of deadlocks. When the demand is non-uniform, we are able to simulate more than 50K drivers without noticing deadlocks.

The average travel time is higher when we compare the same number of drivers in the top and bottom plots of Fig. 3. Take for instance 8K drivers. When the trips are uniformly distributed, the average travel time is as low as 437 (using A*). This changes to around 700 (first point in Fig. 3, bottom left) when trips are according to the non-uniform OD matrix.

In summary, there are many differences between the two kinds of demands. In regular networks deadlocks start to occur for roughly the same number of drivers, independent of type of demand. This is seen in the four plots of Fig. 2 where we were able to simulate, depending on the case, up to 1.5K to 2.5K drivers. In non-regular networks, having regularly distributed trips severely decreases the number of drivers that can use the network because the area with most nodes (generally the center of the city) originates and attracts far too many trips and deadlocks occur there even for a small number of drivers.

5.5 Effect of Traffic Lights

In order to compare the cases with and without use of traffic lights one should look respectively at the left and right sides of figures 2 and 3. In the case with traffic lights, most plots refer to fixed time scheme i.e. cycles are 60 time steps and do not change. Greedy traffic lights did not provide significant reduction in travel times.

The main conclusion is of course that travel times increase due to the delay imposed by the red lights. The magnitude of such delay is different for both networks. In the 6x6 the increase is around 50% (for instance 100

to 150 steps for 500 drivers). There are some differences related to the type of demand with a tendency of less fluctuation when the demand is non-uniform.

In the POA network the variation in travel time increases much more and ranges from 50% to 90% when the demand is uniform (comparing both top plots); when the demand is non-uniform (both bottom plots), it ranges from 100% (20K vehicles) to 170% (40K vehicles).

5.6 Effect of Re-planning

In this section we discuss the effect of using the IR strategy for re-planning. The algorithms used each time a driver re-plans were A* and LPA*. Due to lack of space we report only the former but note that there were no significant differences.

In general one can affirm that re-planning decreases the travel time. However this decrease varies from case to case. Re-planning has a complex effect since it is highly coupled with other factors such as number of drivers, type of demand, regularity of the network, and whether or not traffic lights are employed.

Regarding the number of drivers, as expected, re-planning pays off when the network is close to the saturation level. This can be seen in Fig. 2: in almost all plots (a remarkable exception is top left that corresponds to uniform demand), travel times under re-planning are at least as good as when no re-planning is used. In particular, re-planning yields significantly lower travel times when there are more than 1500 drivers (for example, compare $+$ and $*$ for more than 1500 drivers). This general picture also applies to the POA network (Fig. 3): in almost all cases, and especially for higher number of drivers, replanning yields lower travel times.

It is also remarkable that in some cases re-planning was able to eliminate the deadlock (see both plots at bottom of Fig. 2 and both at top of Fig. 3). This corroborates the intuition that re-planning only pays off when the network is somehow full. Also, re-planning achieves better results when the demand is non-uniform, the network is non-regular, and when traffic lights are active. The fact that these are exactly the conditions re-planning is used in the real-world work as a kind of validation of the agent-based approach.

6 Conclusion and Future Work

In this paper we have presented the latest extensions in the ITSUMO simulator, which aim at provide the users with the tools to design intelligent drivers that can plan and re-plan their routes. To illustrate this we have discussed design issues in two scenarios, stressing differences and similarities found, as well as the effects of type of demand, type of network, number of drivers, efficiency of the routing algorithms, etc.

We are currently working on the experimentation of the variant in which drivers are able to use their individual knowledge (gathered during commuting time) in the selection of route.

REFERENCES

Balmer, M., Meister, K., Rieser, M., Nagel, K., and Axhausen, K. W. (2008). 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*.

Bazzan, A. L. C., de Brito do Amarante, M., Sommer, T., and Benavides, A. J. (2010). ITSUMO: an agent-based simulator for ITS applications. In Rossetti, R., Liu, H., and Tang, S., editors, *Proc. of the 4th Workshop on Artificial Transportation Systems and Simulation*. IEEE.

Bazzan, A. L. C., Nagel, K., and Klügl, F. (2009). Integrating MATSim and ITSUMO for daily replanning under congestion. In *Proceedings of the 35th Latin-American Informatics Conference, CLEI*, Pelotas, Brazil.

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

Klügl, F. and Bazzan, A. L. C. (2004). Route decision behaviour in a commuting scenario. *Journal of Artificial Societies and Social Simulation*, 7(1).

Likhachev, M. (2005). *Search-based Planning for Large Dynamic Environments*. PhD thesis, Carnegie Mellon University.

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

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

Silva, B. C. d., Junges, R., Oliveira, D., and Bazzan, A. L. C. (2006). ITSUMO: an intelligent transportation system for urban mobility. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 1471–1472. ACM Press.

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

Vasirani, M. and Ossowski, S. (2009). Exploring the potential of multiagent learning for autonomous intersection control. In Bazzan, A. L. C. and Klügl, F., editors, *Multi-Agent Systems for Traffic and Transportation*, pages 280–290. IGI Global, Hershey, PA.