Evolving Traffic Scenarios to Test Driver Assistance Systems in Simuations

Torsten Steiner

Fraunhofer ESK,
Hansastrasse 32, 80686 Munich, Germany

Abstract. Nowadays, driver assistance systems involve an ever increasing degree of automatization. Costly effort is put into testing the individual components to ensure proper functioning by the vehicle manufacturers.
However, problems can also arise on a macroscopic scale, as vehicles and infrastructure are recently equipped with short range radio communication ("Car2X"). These problems caused by interaction are of even greater concern than "normal" bugs, as the final product might already have been deployed when the issues first become apparent.
Multi-Agent System (MAS) research refers to such issues as "emergent misbehavior". The said field also brought up an approach to automatically discover the worst consequences of the malfunctions. Hence, a given system under test can already be revised during development, saving a tremendous amount of resources.
The approach from MAS is adapted to the domain of testing driver assistance systems in traffic simulations. A green-light optimal-speed advisory (GLOSA) algorithm is used as an example in which conceptual problems are discovered by the testing system after simpler issues are eliminated.

1 Introduction

Traffic simulations have become a common tool for traffic engineers and advanced driver assistance system (ADAS) developers for testing and validating the performance of their new inventions or infrastructure modifications before deployment. The complexity of ADASs is on the rise due to an ever increasing amount of new sensors and algorithms for automotive applications. This fact already makes it hard for human testers to find all possible causes of errors in common driver assistance systems. However, if several of these systems start to interact (e.g. via Car-to-Car Communication) new use cases can be provided to drivers. As a consequence, several vehicles from different manufacturers will need to communicate and coordinate on future roads - which makes the problem of testing the systems even more difficult.

When performing simulative studies to evaluate the effects of a new ADAS on traffic there are several approaches. A common variant is to build large scenarios, trying to get them as close to reality as possible [14]. Be it by automatic or even manual optimization as in [9]. However, it is often quite hard to aquire

sufficiently accurate detector and traffic signal data to accurately simulate such large scenarios. Hence, [4] built a larger scenario using averaged parameters on a synthetic grid street setup. Another variant is to simulate minimal synthetic scenarios with the sole purpose being the testing of an ADAS with different parameters as in [7], [15].

This paper's contribution is the application of an approach from the area of MAS research [6] to the prototyping of ADAS in traffic simulations. Negative impacts of new systems can automatically be discovered and presented to developers in synthetic small scale scenarios. The said scenarios are automatically generated by a genetic algorithm. This algorithm uses a fitness function that compares the performance of the tested system to the performance of an unmodified system on the same scenario. Consequently, the testing system is able to learn how situations look like in which new ADAS developments lead to bad advice for the drivers. The small scale of the scenarios minimizes the effort to find programming errors or even conceptual problems during development.

To demonstrate the capabilities of this paper's contribution in practice, an instance of the widespread GLOSA application was taken from [7]. Since the testing system found issues in the GLOSA algorithm itself, improvements were implemented in the ADAS logic so that further test runs could be conducted. Finally, the procedure was aborted when a conceptual problem was brought up in a generated scenario. The refactoring required to circumvent that problem would require large changes to the algorithm and new concepts to be introduced, which is beyond the scope of a simple example.

The remainder of this paper is organized as follows: In Section 2 a description of GLOSA and its different implementations is given. The variant from [7] is listed for reference. Section 3 details the concepts of employing a genetic algorithm to find ADAS problems in traffic simulations while Section 4 lists the paramters for the specific tests conducted. Section 5 shows what was achived for the application on the GLOSA demo code. Finally, Section 6 compares the approach from this paper with other similar versions from other fields. Section 7 gives a conclusion and points out interesting new research options.

2 GLOSA in Traffic Simulations

A GLOSA driver assistance system is able to provide the driver with optimal speed recommendations on how to approach a traffic light, so that unnecessary stop-and-go movement is avoided. Consequently, GLOSA systems need to know the traffic signal schedules to provide their recommendations, i.e. the schedule data needs to be transferred to the vehicle in some way. Recent projects and simulations have used dedicated short range communication (DSRC) modules integrated in vehicles and infrastructure [7], [11], [9] as well as crowd sourced data from the drivers smartphones [8].

According to [8], the main benefits of a GLOSA assistance system are threefold: First, the fuel consumption is decreased. Second, traffic flow is smoothed and increased. Third, the environmental impact is decreased. It is also noted,

that given speed recommendations which lie below the current speed of a vehicle do not necessarily increase the total travelling time. The latter is because GLOSA-enabled vehicles cross in the same signal phase as "normal" drivers but are already in motion, as they did not need to halt their vehicles completely.

There are different strategies to achieve the benefits listed above. Some of them can be found in [7], [15], [12], [4] and [3]. To demonstrate the approach used in this paper, the algorithm from [7] was chosen, because of its purely reactive simplicity. It is listed in Figure 1 for convenience.

1: Find the closest traffic light TL
2: Calculate the distance d and time T_{TL} to TL
3: Check phase as T_{TL}
4: **if** GREEN **then**
5: Continue trip
6: Target Speed $U_t = U_{max}$
7: **else if** RED **then**
8: Calculate remaining Red Time T_{red}
9: Calculate target speed for $T_{red} + T_{TL} : U_t$
10: **else if** YELLOW **then**
11: Calculate remaining Yellow Time T_{yellow}
12: Check for possible acceleration
13: Calculate target speed for $T_{yellow} + T_{red} + T_{TL} : U_t$
14: **end if**
15: Advisory speed = $MAX(U_t, U_{min})$ & $MIN(U_t, U_{max})$

Fig. 1. The GLOSA algorithm which was used as an example for problems, that can be found in a given ADAS. Vehicles receive signal phase and timing (SPAT) [5] messages from a traffic signal. From this data, d and T_{TL} can be computed. Based on this computation, recommendations about acceleration or decceleration can be given to the driver. For more details see [7].

3 Evolutionary Testing of ADAS in Traffic Simulations

The main idea of evolutionary testing in ADAS is based on a learning component [6] ("Learner" in Figure 2), that automatically creates different traffic scenarios in which the ADAS under test causes problems. Due to the potentially huge search space (please find an outline of some possibilities in Section 4) a genetic algorithm was chosen for this module.

As a consequence, the general scheme of the testing system is to start on a randomly initialized set of parameters to create a set of traffic scenarios (*generation*) to be evaluated. An evaluation of a single scenario (*individual*) consists of two steps. The first step is to simulate each scenario using an unmodified driver's behavior, while the second step uses the ADAS to influence the behavior. Both steps can of course be parallelized. The output of each step are one or more real

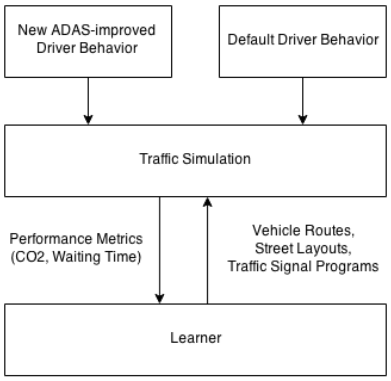

Fig. 2. Overview of the testing approach. This work used a genetic algorithm for the learner and SUMO for the traffic simulation component.

valued numbers on a scale defined by the user (e.g. total CO_2 output or total waiting time). A *fitness function* is then used to incorporate all values from the previous stage and guide the search in the desired direction. *Genetic operators* are then applied to the created scenarios to build a new generation of individuals to be used for the next iteration. After a given number of generations the process is halted.

There are several traffic simulators that could perform the simulations during the evaluation of an individual. For this work, the "Simulation of Urban Mobility" (SUMO) traffic simulator was chosen as it provides enough features to find problems in high level interactions of vehicles on the road. It should be noted that different simulators yield different benefits and that the choice of a simulator greatly depends on the area one wants to find problems in.

3.1 Individual Encoding

SUMO offers a plethora of different configuration parameters to build complex traffic scenarios. As the work presented here only serves as a proof of concept, a choice is needed about which of the parameters to include in the individuals for the genetic algorithm. The employed individual encoding features the following traffic simulation components:

Road Topology A modification of the environment during the evolution is possible, i.e. the layout of streets can be changed in an individual. A design decision was made to use a triangular grid as a base layout, so that more complex intersection layouts and possibilies for vehicles to interact in the simulation can be provided, compared to a grid layout. Consequently, the number of (undirected) streets that meet in an intersection is limited to six, while the presence of each connection on the base graph is controlled by a single bit.

More options open up as each edge in the underlying graph can contain several lanes, each of which can have different turning restrictions. The approach of using a triangle grid compared to e.g. randomly generated junction node locations also avoids the problem of scenarios being regarded as broken by SUMO's parser, as scenarios in which streets cross without an intersection are regarded as invalid.

Traffic Signal Locations and Schedules What is more, it can be determined whether or not a junction should be governed by a traffic signal at all. If the outcome is that there is a signal, the schedules of this signal can also be modified. A constraint was made that limits the search space at this point. Namely, when choosing the signal schedules in a random fashion conflicting streams are a frequent outcome. Since such scenarios do not occur in reality, the interpretation of an individual also does not output them for the simulation.

Traffic Flows Finally, individuals can contain different routes for different vehicle streams, while streams can contain variable numbers of vehicles themselves. The behavior of the vehicles is left to be controlled by user provided applications (i.e. the ADAS under test). Hence, no further information about it is encoded in the individual.

Considering all the possible parameters that make up a traffic simulation there are of course plenty of things which can not be touched in the current version of the presented software system, since this paper only outlines the concept and a simple application. Generally, everything that has not been specifially mentioned before is left to be the default SUMO 0.22 parameter for the given value. Values that are set but left unchanged include the speed limit (set to a static value of 50 km/h for an average urban traffic situation). There are also no accidents and no further obstacles (except for slow vehicles themselves) in the simulations conducted. Nevertheless, it should be noted, that the effort to make parameters or events like the aforementioned ones "reachable" to the genetic algorithm is rather small. When integrating further events, one is basically only limited by the features of the chosen traffic simulator.

3.2 Genetic Operators

The genetic operators employed are twofold. First, a single point mutation can be used to make simple modifications to the topology, i.e. single bit switches determine which of the underlying graph's edges are being used to create the street network later on. Another single point mutation is available to change the timings in traffic signal programs. Second, a crossover operation is available to combine different parts from two parental individuals to create new individuals. This operation takes special care to not "cut" the individuals encoded parameters at the wrong place. This is because swapping parts of traffic signal logic into topology parts would result in invalid scenarios at a later point in the generation process.

Furthermore, special care had to be taken in regard to problems caused by changes to the topology of an individual. For example, scenarios might be changed in a way that takes single segments out of the street grid so that previously generated routes become invalid. This case triggers a repair mechanism in the implementation that fixes the errors: Whenever an existing vehicle's route is broken during a topology mutation it is repaired optimizing for the longest possible route so that interactions between vehicles are maximized.

3.3 Fitness Function

In the same manner as in [6] it was found that an obvious choice for a fitness function fit would be to use the raw difference between one or more metrics of simulation output values for a given individual $i = (topo, sl, ss, flows)$ - where $topo$ is the topology, sl traffic signal locations, ss traffic signal schedules and $flows$ the generated vehicles routes.

For example, a simple fitness function could be $fit_{CO_2}(i) = eval_{ADAS_{CO_2}}(i) - eval_{base_{CO_2}}(i)$, which denotes the total difference between all CO_2 output by all vehicles of a given i, evaluated for the ADAS by the evaluation function $eval_{ADAS_{CO_2}}$ on the one hand and for the basic driver's behavior $eval_{base_{CO_2}}$ on the other hand. In the same manner as for CO_2 one can also use the total waiting time twt as a metric, obtaining a fitness function such as $fit_{twt}(i) = eval_{ADAS_{twt}}(i) - eval_{base_{twt}}(i)$ - where twt represents the time spent by drivers waiting in front of traffic lights or in traffic jams.

As the GLOSA algorithm contains no adaptive elements these simple fitness measures already sufficed to create scenarios in which issues showed up.

4 Applying the Testing Approach to GLOSA Traffic Simulations

To apply the testing system to drivers behaving according to the algorithm from Figure 1 some more assumptions and settings were made, which the following paragraphs are going to describe.

Basically, the initial implementation of the GLOSA logic was kept as close as possible to [7]. That means, that the SUMO traffic simulator was used as a base for the simulations conducted. Consequently, its traffic flow model [10] is used as a fallback, as soon as the GLOSA algorithm does not compute any recommendation. This might be the case when there is no traffic light on the upcoming route, or the next traffic light is too far away to be in radio range. Also, the prototypical ADAS is connected to SUMO via the TRACI interface, so vehicles in the simulation can be given GLOSA speed recommendations.

To keep things simple, random influcences that do not lead to the test goal were kept out of the simulations. Concretely, the evaluation runs were done without further simulator-coupling (e.g. a network simulator was not used), as the point of applying the testing approach is to find negative highlevel interactions which are caused by following the final ADAS advice. Hence, it is assumed that

the vehicles "know" the traffic signal schedules and can compute their recommended speed accordingly (i.e. the communication links always work). What is more, the drivers always accept the GLOSA recommendations as long as there are no collisions involved. This means that slower vehicles in front are not overtaken. However, it is perfectly possible to build more complex simulation setups in the future to provide for even larger possibilies of failure from different sources.

Regarding the configuration of the genetic algorithm the following settings were made: The street layout used in the experimental runs was built based on a quite small number (2) of triangles to construct the concrete SUMO scenario on. The number of triangles was chosen, so that a small street net is produced which guarantees a quick overview of what is happening to the developer.

The number of lanes of a street coming into an intersection was configured to be between zero (no connection to a junction at all) and four. Note that this refers to the number of lanes in one direction on a single street - consequently, there can be a total of eight lanes on a single street on a given intersection.

Traffic signal schedules were created to contain values between three and 64 seconds for a phase.

The vehicles routes were made modifiable in different ranges for the application of the testing system to the given GLOSA algorithm. First, a series of runs was conducted with only a single vehicle in the simulation, so that simple implementation issues (such as parsing or interpreting SUMO's TRACI output incorrectly) could be resolved quickly. Afterwards, the genetic algorithm was allowed to insert between two and four vehicle streams into the traffic scenario. Each of these streams could contain between three and 15 vehicles.

Traffic scenarios were not aborted but simulated until every inserted vehicle had reached its final destination, i.e. it left the simulation. Hence, the number of simulated timesteps variied from scenario to scenario and was only indirectly influenced by the learning system via the aforementioned parameters.

Finally, fit_{twt} was used as a fitness function, so that speed modulation had no impact on an individual's fitness as it would have been the case when using fit_{CO_2}. This choice was made because the learning system would run into scenarios in which vehicles crossed the stop line just as a phase switch occurred. Consequently, the GLOSA algorithm would always oscillate between break and acceleration state, raising the total CO_2 output of an individual. This was regarded as a minor detail to be fixed, compared to the conceptual issues described later.

5 Results

To conduct the experimental runs, the genetic algorithm was configured as described in the previous section. The ADAS logic used to compute the GLOSA recommendations was taken from [7] and is given in Figure 1 for reference.

# flows	Average	Maximum
2	27.95	105.86
3	7.06	40.29
4	2.07	3.83

Table 1. Results for the testing system's efficiency. "Average" denotes the average of all values of $\frac{ADAS_{twt}}{base_{twt}}$ over a series of runs, while "Maximum" stands for the result of the best run in a series. See Section 3.3 for the according definitions.

5.1 Quantitative Results

Quantitative results are listed in Table 1. To obtain the data for each row a series of ten runs was conducted to account for random effects of the genetic algorithm. What is more, the number of generations used was always 20, while the number of individuals per generation was kept at 24. There were always two vehicles in a flow. Since the combined total waiting time of all vehicles in a traffic scenario can be zero, those scenarios were filtered from the testing system's output, so that relative increases could be computed. It is striking that the testing system could generate scenarios with large losses in traffic efficiency. The main cause for inefficiencies is described in the next section. It should be noted that the runs conducted were done using the improved version of the GLOSA algorithm from [7] (i.e. a version including all the changes described in the next section).

5.2 Qualitative Results - Improving on a Reactive Algorithm

When the approach was applied to practical simulations, the first problems found were simple programming errors as they were mentioned before. The general process here was to evolve simulation scenarios in which such errors occurred, fix the corresponding problem and introduce the scenario as a testcase for the ADAS logic itself. The addition of an evolved scenario as a testcase was also kept as the errors got more complex in later runs, since it prevented to reintroduce problems into the system which were already fixed before. For the early stage of development only a single vehicle was used for the simulations. Only after the search did not yield any scenario in which the GLOSA logic performed worse than SUMO's basic driver behavior the genetic algorithm was allowed to use more than one vehicle.

After these first trivial mistakes were fixed, two problems in the algorithm from Figure 1 were found by the genetic algorithm. When looking at those scenarios manually, both of them displayed situations in which vehicles would drive remarkably slow. Looking at the code paths used by these slow vehicles in the GLOSA algorithm exposed the root of the problem to be in line three of the algorithm. More concretely, the time for a given vehicle to reach the next traffic light T_{TL} was computed by Equation (1). In the equation, d is the distance to the traffic signal, v the vehicle's current velocity and a the vehicle's current acceleration.

$$T_{TL} = \begin{cases} \frac{d}{v}, & \text{when } a = 0 \\ -\frac{v}{a} + \sqrt{\frac{v^2}{a^2} + \frac{2d}{a}}, & \text{when } a \neq 0 \end{cases} \quad (1)$$

With a scenario pointing at the problem is was obvious to see, that the given GLOSA algorithm did not cover the case of $a = 0$, while $v \neq 0$ (and the speed limit allowed a higher velocity v_{max}). More concretely, if the aforementioned case was hit, T_{TL} was set to an incorrect value as a consequence and the remainder of the algorithm would only output incorrect values. To correct the error, the computation of T_{TL} was extended for that case to be as given in (2) with $t_{v_{max}} = \frac{v_{max} - v_0}{a}$

$$T_{TL} = t_{v_{max}} + \frac{d - (v_0 * t_{v_{max}} + 1/2 * a_{max} * t_{v_{max}}^2)}{v_{max}}, \quad (2)$$

After the above computation was integrated the next iteration of the test system brought up a similar issue. Again, the error location was in the T_{TL} computation and hence provided incorrect input for the remaining algorithm. Since [7] used the current acceleration to compute T_{TL} the case of a possible stronger acceleration is disregarded. To improve the algorithm the maximal acceleration of the given vehicle was used instead changing $t_{v_{max}}$ in Equation 2 to be $\frac{v_{max} - v_0}{a_{max}}$, with a_{max} being the vehicle's maximum acceleration.

5.3 Conceptual Problems Detected

Finally, after all problems from the previous section were fixed as described, the genetic algorithm ended up creating a scenario in which a conceptual problem of the employed GLOSA approach became apparent. The scenario is depicted in Figure 3. One can see a number of vehicles closing in on an intersection. Vehicle number (1) moves very slowly because its assistance system hands out advice to do so, knowing that the upper signal will turn green in some seconds. However, the following vehicles (2-4) are blocked by vehicle (1). As a consequence, (2-4) can not pass the junction as it would have been possible had the GLOSA system not been in place in (1).

This is the point at which the decision was made to interrupt the test runs on the GLOSA algorithm. It became apparent that there are more elaborate concepts needed to resolve the situation, so that the overall waiting time and CO_2 output are still kept to a minimum. It should be noted that the problem is already known in the literature, e.g. [11] found the situation and recommends the usage of GLOSA only for "simple" intersections without overlapping lanes. However, future coordination mechanisms could certainly also be used to resolve such situations, so that GLOSA can also be provided for more complex intersections. For example, vehicles could dynamically negotiate a solution while approaching a traffic light.

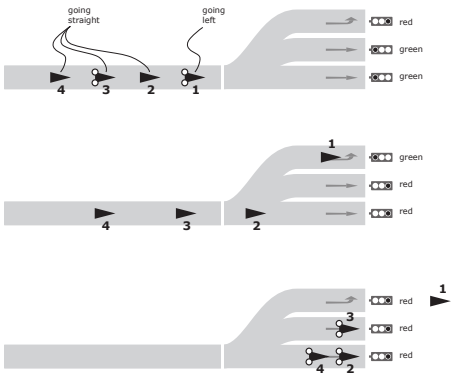

Fig. 3. The conceptual problem discovered in the GLOSA algorithm from [7]. The snapshots depict a situation at a traffic signal. Time flows from top to bottom. Vehicle (1) is slow due to GLOSA advice and blocks (2), (3) and (4). Consequently, three vehicles are forced into stop-and-go movement while only a single one can drive efficiently.

6 Related Work

As mentioned before the initial idea for the creation of the presented testing system was taken from [6] which applied the general testing approach to the domain of the Pickup and Delivery Problem. Later, the approach was applied to attack vehicular ad hoc networks [2] which comes closest to the work presented in this paper. However, the focus of [2] is on mobile networks, which are being attacked by malicious agents while this work concentrates on effects caused by bad advice given to drivers via assistance systems. What is more, the environment is never modified as a part of the simulation.

Another approach that comes close to the work presented here was shown in [1], however the domain of application were air traffic scenarios. Consequently, the environment was spatially unconstrained (airspace) except for other planes. On top of that, the goal was to find scenarios with a high complexity as the algorithms were expected to perform poorly under these conditions. Also, (and in the same manner as [13]) the employed optimization function was not crafted to find misbehaviors by using another given system.

When it comes to ADAS evaluations there is a general issue of how to pick a good simulation scenario layout for the evaluation of new systems/algorithms. The problem is that scenarios need to be specific enough to show the benefit of a given system, while they also need to be general enough to demo that the given system also works under other conditions. Generally, there are three kinds of simulation studies for new assistance systems used in the literature. First, using a small number of very small synthetic scenarios [7], [13]. Second, a few realistic scenarios [11], [14] or third, a grid street layout to base their evaluations on as in

[4]. Studies that decide for realistic scenarios mostly have to deal with the bad availability of traffic flow or traffic signal schedule data.

However, there seems to be no general approach that maps the search technique to traffic simulations as presented in this paper. Scenarios are always static and no search is performed.

7 Conclusion

This paper has presented the application of a semiautomatic testing process for driver assistance systems. A simple GLOSA algorithm was taken as an example to show what kind of issues the testing system can reveal in practice. It was shown that there is a conceptual problem in the purely reactive GLOSA algorithm from [7]. A driver who follows correct GLOSA advice can block other drivers from crossing a traffic signal at green in a situation as shown in Figure 3.

It was proposed to develop a Car2Car approach to fix the problem, as it was shown that a simple broadcast of traffic light phase and timing (SPAT) [5] is not enough for every situation. This would make more complex coordination mechanisms a necessity. Work on the introduction of such mechanisms has only started recently and can hopefully profit from the system presented here, as the approach also works for more adaptive systems when the fitness function is changed accordingly.

Future extensions and possible applications of the presented test system are manifold. The testing system was not employed to its maximum capabilities, since the GLOSA algorithm was purely reactive. Hence, the application to systems like [13] or the aforementioned Car2Car coordination mechanism would be especially interesting, since [6] already showed how self-adaption in systems can be automatically exploited by the search algorithm.

Also, the application of the principle is not limited to the comparision to a simulators basic driver model. The basic model can easily be swapped for another ADAS. This variant could be used to improve on a working assistance system when adding new features, while making sure not to break performance in the process.

What is more, one could deploy several vehicles with different ADASs in a single scenario. The test system in its presented form is applicable to all ADASs that change a driver's behavior. Hence, it would be simple to also use it on simulations in which several different ADASs interact. Unforeseen interactions could be brought up and fixed before they might happen in reality. This point is expected to gain further importance in the near future as a lot of manufacturers are focussing on the development of autonomous vehicles.

References

1. Alam, S., Shafi, K., Abbass, H., Barlow, M.: Evolving Air Traffic Scenarios for the Evaluation of Conflict Detection Models. 6th Eurocontrol Innovation Research Workshop and Conference, Eurocontrol Experiment Research Centre pp. 1–8 (2007)

2. Bergmann, K.: Vulnerability Testing In Wireless Ad-hoc Networks Using Incremental Adaptive Corrective Learning. Dissertation, University of Calgary (2014), http://theses.ucalgary.ca/handle/11023/1504
3. Chao-Qun, M., Hai-Jun, H., Tie-Qiao, T.: Improving Urban Traffic by Velocity Guidance. In: 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA). vol. 2, pp. 383–387. IEEE (Oct 2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4659788
4. Eckhoff, D., Halmos, B., German, R.: Potentials and limitations of Green Light Optimal Speed Advisory systems. VNC (Section IV) (2013), http://www7old.informatik.uni-erlangen.de/ eckhoff/publications/pdf/eckhoff2013potentials.pdf
5. ETSI: TS 102 894-1 - V1.1.1 - Intelligent Transport Systems (ITS); Users and applications requirements; Part 1: Facility layer structure, functional requirements and specifications 1, 1–56 (2013)
6. Hudson, J., Denzinger, J., Kasinger, H., Bauer, B.: Efficiency Testing of Self-Adapting Systems by Learning of Event Sequences. In: ADAPTIVE-10. pp. 200–205. No. c (2010)
7. Katsaros, K., Kernchen, R., Dianati, M., Rieck, D.: Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform. In: 2011 7th International Wireless Communications and Mobile Computing Conference. pp. 918–923. IEEE (Jul 2011)
8. Koukoumidis, E., Peh, L., Martonosi, M.: SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advisory. In: Proceedings of the 9th international conference on Mobile systems, applications, and services. vol. June 28-Ju, pp. 127–140. ACM (2011)
9. Krajzewicz, D., Bieker, L., Erdmann, J.: Preparing Simulative Evaluation of the GLOSA Application. elib.dlr.de (October), 1–11 (2012), http://elib.dlr.de/78905/1/ITSW2012_GLOSA.pdf
10. Krauß, S.: Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. D L R - Forschungsberichte (1998)
11. Niebel, W.: Cost-Benefit-Based Implementation Strategy for Green Light Optimised Speed Advisory (GLOSA). Activities of Transport Telematics 2013(C), 312–320 (2013), http://link.springer.com/chapter/10.1007/978-3-642-41647-7_38
12. Sanchez, M., Cano, J.c., Kim, D.: Predicting Traffic lights to Improve Urban Traffic Fuel Consumption. In: 2006 6th International Conference on ITS Telecommunications. pp. 331–336. IEEE (Jun 2006)
13. Seredynski, M., Mazurczyk, W., Khadraoui, D.: Multi-segment Green Light Optimal Speed Advisory. 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum pp. 459–465 (May 2013)
14. Sommer, C., Eckhoff, D., Dressler, F.: Improving the Accuracy of IVC Simulation Using Crowd-sourced Geodata. PIK - Praxis der Informationsverarbeitung und Kommunikation 33(4), 278–283 (Jan 2010), http://www.degruyter.com/view/j/piko.2010.33.issue-4/piko.2010.047/piko.2010.047.xml
15. Wegener, A., Hellbruck, H., Wewetzer, C., Lubke, A.: VANET Simulation Environment with Feedback Loop and its Application to Traffic Light Assistance. In: 2008 IEEE Globecom Workshops. pp. 1–7. IEEE (Nov 2008)