TraffSim - A Traffic Simulator for Investigations of Congestion Minimization through Dynamic Vehicle Rerouting

Christian Backfrieder*, Gerald Ostermayer* and Christoph F. Mecklenbräuker†

*Research Group Networks and Mobility, UAS Upper Austria, Hagenberg, Austria
Email: christian.backfrieder@fh-hagenberg.at, gerald.ostermayer@fh-hagenberg.at
†Christian Doppler Lab Wireless Technology for Sustainable Mobility
Vienna University of Technology, Vienna, Austria
Email: cfm@nt.tuwien.ac.at

Abstract—With our interest to reduce traffic congestion, simulations are necessary to investigate how to achieve this goal. Because of the difficulty of real-time experiments, traffic simulators are a good and widely used tool for analyzing specific situations of the traffic flow virtually. This paper introduces TraffSim, a new platform-independent framework, which is capable of simulating microscopic vehicular road traffic in a time-continuous manner. It is able to simulate scenarios close to reality by using OpenStreetMap but also artificial road networks. TraffSim is able to represent multilane motorways or dense city road networks including multiple types of intersections or one-way. Apart from comprehensive support for models for longitudinal movement, lane-changing and fuel-consumption, a supplementary component combines congestion detection, prediction and minimization of traffic jams through intelligent rerouting algorithms. We present the architectural and logical aspects of the simulator, as well as the user interface and statistical features.

Keywords—road traffic, traffic simulation, jam, congestion, traffic management, dynamic rerouting, time-continuous simulation, intersection, open street map

I. INTRODUCTION

Modeling of vehicular road traffic is a very complex topic. It has serious consequences for human drivers, residents of roads, animals and the environment how we use vehicles to travel from A to B. The volume of traffic increases steadily, especially in bigger cities and areas of high population density. Therefore, it is essential to deal with the big amount of vehicles optimally to minimize the driving time, traveled distance and subsequently exhausted emissions. Traffic jams cause, in addition to the mentioned issues, huge public costs for cities or countries, and should therefore be reduced. For that reasons it is important to learn and understand how to deal with this problem, and to achieve an improvement by intelligent control and management of traffic.

In order to investigate the influence of traffic management, simulators are a practical instrument for predicting the impacts of different deployed applications. Different situations and parameters can be put into them with the objective of prognosticate possible implications resulting of the input change. Simulators can be mainly classified in microscopic or macroscopic types, depending on the granularity of investigation and temporal resolution. Macroscopic models deal with the traffic state as a whole with attributes like traffic flow, density or average speed, while microscopic models analyze the single vehicles as single particles in the system. Each simulated unit has its counterpart in reality. The TraffSim simulation framework is a microscopic simulator, calculating speed, acceleration, fuel consumption, travel distance or time among other parameters for each modeled vehicle separately.

TraffSim was developed with the idea of an extensible, freely available simulation framework, which is fully configurable in advance by XML input files. In order to investigate and learn from the impacts of different parameter changes, it is very important to allow reproducibility in simulations. TraffSim is able to re-run a simulation as desired and creates equal output when using the same input configuration, assuming not to use parallel calculations in a single simulation, because in this case the operating system scheduler influences the order of execution and thus also the results. The input configuration includes definition of the road network, intersection control, vehicle types, car-following models for longitudinal movement, lane-change models and routes to define how vehicles intend to travel to their target. Furthermore, fuel-consumption is calculated to investigate the emissions exhausted by the vehicles.

Apart from the basic traffic simulation framework, TraffSim also contains a congestion detection and rerouting component. With the help of this, traffic jams can be assessed, vehicles which are likely to pass the jammed segments are identified and their routes are re-calculated, so that the congestion can be avoided or at least the severity of the jam can be decreased.

The rest of the paper is organized as follows. In the next section, comparable projects from literature are elaborated and requirements for TraffSim are defined. The third section deals with the basic framework itself, and introduces the Java architecture of the simulator as well as implemented models. Furthermore, the Eclipse RCP user interface is presented in detail. Section IV elaborates the congestion detection...
component as well as re-routing algorithms. In the fifth section, applications of the simulator are explained. Section VI concludes the paper and gives an overview of planned future development.

II. RELATED WORK

Several projects dealing with traffic simulation exist in literature, which all have some strengths in a specific field, and also differ in some aspects. The following major traffic simulation projects were selected, and we want to give a short overview about the features and qualities of them. Additionally, some

A. Existing Projects

A very powerful simulator is MovSim [1], which was initiated at the TU Dresden as platform independent Java open-source project. It is a microscopic simulator, which has its strengths in having implemented several different models for car-following, using its own fuel-consumption model and lane change model MOBIL (discussed in [2]). Nevertheless, it has no importer for OpenStreetMap (OSM) networks, and also no support for intersections (meaning not motorway ramps, but intersections with multiple in- and output lanes).

Another huge microscopic and multi-modal traffic flow simulation framework is VISSIM, which is developed by PTV Planung Transport Verkehr AG in Karlsruhe, Germany. It is used worldwide for microscopic traffic simulations in cities and able to simulate different vehicle types, but also public transport, pedestrians and bicyclists. VISSIM is running under windows and licensed commercially.

The Traffic Simulation Framework explained in [3] is a cellular automaton (CA)-based simulation model, which further is an extension of the Nagel-Schreckenberg model [4]. It is implemented for Windows platforms using the C# programming language. CA simulators split the physical location into cells of length $\Delta x$, which can hold the state occupied or free for each time stamp. In addition, more attributes such as speed can be assigned to occupied cells. They are often faster than continuous simulators, but due to the fact that no fuel consumption model is implemented in this framework and CA models often lead to unrealistic values or unpredictable outcomes, which are among other things a result of stochastic terms [5], it does not fit the needs defined for TraffSim.

A further microscopic and macroscopic free-flow traffic simulator is FreeSim [6]. It is platform independent and describes itself as being easily extensible. FreeSim models vehicles basically as self-managed entities, which can send and receive information via a central server and traverse along freeway segments. It allows monitoring of travel times and congestion through the centralized server, but is mainly designed for wide-ranging motorway networks rather than dense cities.

B. Requirements for TraffSim

Probably the most comprehensive simulator with free availability is the open-source project SUMO (Simulation of Urban Mobility) in [7]. It supports definition of several types of vehicle models (longitudinal, lane-change etc.). Also, a road intersection model exists that is able to supervise signal-controlled intersections as well as prioritized intersections or others without prioritization [8]. Nevertheless, the intersection model for SUMO does not yet support stop signs. Besides the basic SUMO simulator framework including source code, quite a few projects extend SUMO to fit their own needs and provide components for certain tasks.

In the field of intelligent traffic management and dynamic rerouting one can also find projects in literature. An extensive project dealing mainly with vehicle rerouting strategies is presented in [9] and [10]. It also uses simulations with SUMO simulator for analyzing the rerouting behavior and creates statements about savings in time or traveled distance. However, this results do not contain any values or figures about fuel consumption and savings due to the optimization. Furthermore, no possibility for prediction of traffic emergence is included within this project.

Several dealing with vehicle routing exist. The authors of [11] focus on crash reduction through rerouting using vehicle to vehicle communication through a VISSIM simulation. Among others, optimizing vehicles’ routes for lower travel time using SUMO is proposed in [12], [13] and [14]. Additionally to the functionality of existing projects, TraffSim combines the advantages and also provides forecasting mechanisms for further optimizing routes and reveal new possibilities for traffic flow optimization.

Also, theoretical models using Dynamic Traffic Assignment (DTA) can be found in [15] and [16]. In contrast to live rerouting, this method iteratively recalculates the optimal assignment of routes until either system optimum (minimize total travel time of all vehicles) or user equilibrium (best balance for travel time) [17] is achieved. This is a good possibility for comparing results from live simulations, but nevertheless is out of scope of this work.

B. Requirements for TraffSim

TraffSim was implemented for investigating a defined topic within the research project SmartTraffic, executed at the UAS Upper Austria. Its main scope is to find out how intelligent management of traffic flows can reduce congestion and traffic jams. This is done by assuming a working communication between the cars and infrastructure, which is then used to smartly re-route cars to bypass possible traffic jams. For this to work congestion of road segments must be detected and predicted reliably. Furthermore, in case of a detected congestion the affected vehicles have to be identified and considered for using an alternative route to their original destination which bypasses the congestion. However, the rerouting mechanism has to take care not to shift the congestion from one area to another, but intelligently calculate new
routes and balance the arising traffic amount. The gained benefits from the fast and early information propagation, compared to radio broadcast as well as the performance of the congestion detection and re-routing mechanism, can be measured for example in less fuel consumption, less time consumption or lower CO₂ emissions.

For achieving this goal, the following requirements were defined for TraffSim:

- fully reproducible results
- realistic longitudinal models
- fuel consumption model
- usage of real road networks (OSM)
- cleanly encapsulated architecture
  - full abstraction of the features
  - well defined interfaces
  - possibility for extensions without any kernel code changes
- interactive control for developer’s self-learning during implementation and fact-finding phase
- modeling of intersections
- crash detection
- time and date attributes for modeling peak hours or other time-dependent situations
- time and location-continuous simulation for precise results
- platform independent
- parallel simulations of multiple scenarios
- performance optimized through multi-threading
- efficient re-routing algorithms
- dynamic congestion detection
- congestion prediction heuristics
- option for parallel execution within a single simulation
- free availability without any restrictions

It should be noted that fully equal results cannot be guaranteed when using parallel calculations within a single simulation obviously, because the order of executions of single tasks is the responsibility of the operating systems scheduler. This means that parallel calculation of all tasks within a single time step (e.g. longitudinal propagation of vehicles, lane change decisions, ...) does indeed speed up the simulation enormously, but simulations with equal input do not produce the very same results. However, the output is still very similar, so that this small difference can be accepted because acceleration of the simulation outperforms the minimal variation.

All the aggregated functions provided by TraffSim are necessary to perform the investigations planned within the SmartTraffic research project. None of the described simulators includes those required functions without limitations. TraffSim combines the features and strengths of existing simulators which are required for fulfilling these defined requirements. It further extends functionality by routing and congestion detection possibilities or implementations of intersection logic.

### III. The TraffSim Framework

#### A. Environment

This application is implemented as Eclipse Rich Client Platform (RCP) application, which brings several benefits for both developers and users. First of all, it is able to deploy to a variety of desktop operating systems, such as Windows, Linux and Mac OS X. It provides an automated update mechanism, is built up through a plug-in framework, which easily enables adding or exchanging components and uses the commonly known Eclipse UI, including the view concept, preference screen or toolbars [18]. This is certainly a benefit for users which are already familiar with other Eclipse products as well. The user interface and business logic is separated from each other, so the described components are implemented in pure Java.

#### B. Architecture and Components

The modules of TraffSim can be divided as shown in the component diagram in figure 1.

The Kernel is the central manager of the application, which creates multiple simulation models. These hold all necessary information about a simulation and read the configuration from the input files. The crash detector, statistics component, re-router, congestion detector and all required models are set up by this component as defined in the configuration. TraffSim is a time- and location continuous simulator, which means that vehicle positions are available for any arbitrary time stamp. The simulation runner is responsible for driving the simulation forward by a configurable interval. This mechanism is implemented event-based, meaning that all components interested in time updates have to be registered within the simulation runner, and get subsequently notified about updates. They do their job, and wait for the next update. In parallel mode, the tasks within a single time step are executed parallelly, as long as they are independent from each other. Examples for task sets of parallel execution are:

- calculation of accelerations of all vehicles
- position updates for all vehicles

![Figure 1. TraffSim architecture](image-url)
re-calculation of the vehicles’ routes
• updates of dynamic graph weights

For components without any necessity to be updated with such a high frequency as the re-router or statistics component, the desired update interval can be configured within TraffSim. It is ensured that the simulator sticks to this defined interval even if a single step takes more time than this by simply waiting until computation is finished before continuing to update the simulation time.

The user interface (UI) component also shows separate windows for each created simulation, besides displaying an environment container for the simulations where users can interact and for example load existing configurations, show, hide or move views, display details about any entities or pause a running simulation.

So as to optimize performance, concurrent execution in multiple Java threads is used. This asynchronous execution has also disadvantages in requiring more effort for synchronizing threads and keeping commonly used data consistent. However, this effort arises in the development phase and does not influence the user of the simulator.

Furthermore, the UI in TraffSim is implemented asynchronously in order not to block the running simulation while updating. It paints the current state as often as possible. Some statistic graphs can be updated continuously, which is also executed in parallel to the main worker. The worst case when having too less CPU power is a lower UI frame rate, which can be avoided by increasing sleep time between simulation time updates. The crash detector can be configured to run asynchronous as well, however, this is not recommended in simulations which should produce scientific results because there is a low probability for overlooking crashes, but only during testing and developing phase.

In consequence of the abstraction and separation of the components it is also possible to run multiple simulations in parallel and gain knowledge about impacts of parameter changes visually. Each simulation is executed in a sandbox and has its own environment without any mutual manipulation, as illustrated in figure 2.

C. Data Structure

Moving entities in TraffSim are called vehicles, which can be set up individually to represent small cars, estate cars or trucks by freely configurable lengths, widths, various models or engine.

The static infrastructure is reproduced virtually through road segments, which is a part of a street in a single direction and owns its geometry, a speed limit and an ID for routing. It supplementary contains lane segments, which correspond to the number of lanes in one direction. The third main data type is an intersection, which connects two or more road segments. It also contains lane connectors, which connect each lane inwards to each possible lane outwards and define the vehicle’s tracks through the intersection, as depicted in figure 3.

D. Input Structure

Each simulation uses multiple XML files for defining all necessary input parameters. These are referenced in a main configuration file, which is can be loaded within the UI by the user. An example for a definition of a vehicle is available in listing 1.

```xml
<Vehicle id="1">
    <label></label>
    <vehicleType>Car</vehicleType>
    <length>4.77</length>
    <width>1.82</width>
    <laneChaneModel>MOBIL1</laneChaneModel>
    <longitudinalModel>ACC1</longitudinalModel>
    <consumptionModel>Diesel_90kw</consumptionModel>
    <routeId>127</routeId>
    <startDate>2013-10-10 15:21:19.216 UTC</startDate>
    <initialSpeed>20.0</initialSpeed>
</Vehicle>
```

Listing 1. Vehicle configuration

E. Road Network

1) Road Segments: The road network is stored in a similar XML format to vehicles. It contains OSM geometry data, augmented with lane geometry and right and left edges of the roads for nice visual presentation. TraffSim allows smoothing of very sharp curves and creation of a specified number of lanes by shifting the original geometry by the lane width and support for entry and exit lanes. The mentioned road segments are connectors between OSM nodes, which are road crossings. Of course, intersections are areas rather than simple points where lines are connected. Therefore, road segments are shortened by the intersection radius and assigned connected to the intersection. OSM data often contains very short road segments, which are removed if they disappear during the shortening procedure, and of course all routes referencing...
them are also updated. Figure 4 shows the network before and after TraffSim’s postprocessing.

2) Intersections: Road crossings are modeled as separate entities, which handle all vehicles approaching, waiting, inside and outside, but already granted to enter the intersection. The logic is encapsulated by an abstract base class, which specifies the interface for implementing objects. They provide an interface method to determine whether or not a specific vehicle $v$ can enter the intersection from lane $L_1$ inwards to lane $L_2$ outwards. The intersection implementation itself has to lead the vehicles through in a way close to reality and avoid crashes.

TraffSim contains a comprehensive set of intersection types, trying to represent as much situations of reality as possible. These are unregulated intersections without prioritization, regulated intersections with road signs or traffic lights and roundabouts, as shown in figure 5.

A complex logic is necessary in order to provide a realistic and smooth drive-through while at the same time avoid deadlocks and accidents due to erroneous right of way decisions. Additionally, the logic is organized decentrally and basically independent from other intersections to allow parallel entry requests from multiple vehicles and parallel evaluations for multiple intersections. For an extensive specification and description of the intersection logic, it is referred to [19], which contains TraffSim’s intersection control algorithm in great detail.

3) OSM Import: TraffSim can use freely available real road networks from OpenStreetMap (OSM) with arbitrary size but also custom designed networks. Import is possible from already downloaded OSM XML or PBF files [20]. A wizard (figure 6) also allows direct selection of an area on a map, which then is downloaded and saved as OSM file, as shown in figure 7.

Maps from OSM can be modified by specific editors, such as the platform independent JOSM [21]. Moreover, it is possible to enhance the maps with meta information which is essential for intersections like the number of lanes including their direction restrictions or details about the regulation type as traffic lights or signs and right of way rules. This meta information is partly already available online, if not it can be edited and enhanced with JOSM for usage within TraffSim. However, we use as much available information as possible from online sources to keep configuration effort minimal. In cases where no information is available, TraffSim provides an interface for configuration of the intersections and other metadata and persisting the setup [19].
F. Models

1) Longitudinal Models: Those models deal with the vehicles propagation along the road. Within TraffSim, they are implemented as time continuous car-following models, according to [22]. To be more precise, the current simulator includes an intelligent driver model (IDM) and a model for adaptive cruise control (ACC), which is similar to automated driving enriched with some human aspects.

Those models must be able to calculate the acceleration for a given vehicle on a given lane of the road, including important attributes as current speed, current acceleration, speed of front vehicle and speed limit.

2) Fuel-Consumption Models: Basically, fuel consumption is a very good argument when talking about saving costs as a result of intelligent traffic management, not only because it also implies a reduction of emissions. Therefore, the simulation framework must provide methods to calculate the instantaneous fuel consumption, as well as summing it up and relate it with traveled time and kilometers.

TraffSim provides an implementation of the microscopic model, which uses the current speed and acceleration values as dynamic input. Aside from that, it uses engine attributes as the cylinder volume, power of the engine, idle and maximum rotation rates and gear ratios as well as physical car data like the car mass, cross section surface or the dynamic tire radius for the calculation. A very detailed explanation of this model can be found in [25] and references therein.

3) Lane-Change Models: For roads with multiple lanes a model for the decision of lane changes is essential in a simulator. It is anything but trivial to develop a model which is crash-free and is considerate towards other vehicles politely and realistic at the same time. During research, we had a deeper look in the model used in [1], and it fits the requirements of TraffSim sufficiently. The general model is called "Minimizing Overall Braking Induced by Lane Change (MOBIL)" and elucidated in [2]. This model does not only take the risk for the lane-changing vehicle into account, but also considers effects of neighbor vehicles such as the old and new follower or old and new leader in terms of longitudinal accelerations, calculated by the assigned car-following model. Thus, it is also possible to perform lane changes for allowing an entering vehicle to leave the ramp and merge into the continuing lane.

We extended this lane change model in order to be also applicable for lane changes which are necessary due to multiple lanes on the current road segment and only a single lane leading to the desired target segment. This can be entry/exit lanes on a highway as figure 8 shows, or lanes with restrictions allowing only turns to a single direction. Vehicles must ensure to be on the right lane before the segment ends to avoid driving in the wrong direction.

Especially in congested situations with convoys of vehicles moving towards a separation of ways, the lane change is often not possible without the help of other drivers. Vehicles on the target lane need to leave enough space in front of them in order to let the lane-changing vehicle enter its target lane. If not doing so, the lane change is not possible due to occupation of the target lane by vehicles driving through, and vehicles need to stop suddenly at the end of the segment waiting for a possible gap on the target lane, which is rather unrealistic. Therefore, the longitudinal function of neighbored vehicles is influenced as soon as the urgency of the required lane change reaches a certain threshold, where 0.1 turned out to be a good value. The urgency value \( U_{\text{lanechange}} \) is calculated according to equations 1. The remaining distance to the intersection is \( d_{\text{intersection}} \), the distance of the last road segment before the intersection is \( d_{\text{lastSeg}} \). We subtract 20 percent of this distance from the overall distance because the lane change should be completed before the vehicle reaches the end of the road segment. \( d_{\text{decision}} \) represents the distance before the branch point where a vehicle starts to consider a lane change to its target lane, which is calculated by using a defined time \( t_{\text{lanechange}} \) before the branch multiplied by the current road speed limit \( v_{\text{limit}} \).

$$
U_{\text{lanechange}} = 1 - \frac{(d_{\text{intersection}} - d_{\text{lastSeg}} \times 0.2)}{d_{\text{decision}}} \\
\text{d}_{\text{decision}} = t_{\text{lanechange}} \times \frac{1}{v_{\text{limit}}} \tag{1}
$$

The calculated urgency factor in the range \([0, 1] \) is also used for the slow-down of vehicles which are on the neighbor lane. It is used as percentage of slow-down due to a pending lane change, as equation 2 describes. The current speed limit \( v_{\text{limit}} \) is modified by the urgency \( U_{\text{neighbor}} \), so that the calculated target speed \( v_{\text{target}} \) of the neighbor vehicle is adjusted. It slows down until a safe lane change is possible.

$$
v_{\text{target}} = v_{\text{limit}} \times U_{\text{neighbor}} \tag{2}
$$

G. Statistics

A very important functionality is the recording of interesting data for analysis after finished simulations. TraffSim records data for vehicles, roads and intersections automatically for live display or postprocessing, which includes speed, acceleration, overall traveled distance and fuel consumption for vehicles and the vehicle throughput including time, vehicle id, type and speed for each road segment and intersection.
In large simulation scenarios, the amount of emerging data is rather huge, up to a few gigabytes. Thus, TraffSim serializes this data as Java value objects in a defined interval to safe central memory. In order to allow efficient post-processing of data, TraffSim converts the serialized data to MATLAB mat files [23].

H. Routing

Vehicle routing is a very important topic for TraffSim, since scenarios are intended to be reproduced with different parameter sets, but the same initial routes for the defined vehicles. To be exact, each vehicle has its predefined source, target and route from source to target. For the intersection controls described in III-E2 to work properly, vehicles must know where they would like to go when asking an intersection for permission to enter. The intersection accordingly leads the vehicle to the desired connector to continue its route. Generally, a route is a list of ids which are to be passed by a vehicle to reach its destination. Each road segment owns a routing id. The routes are generated prior to simulation by external tools and saved within the TraffSim configuration. The framework provides a basic implementation for route creation, which is very simple. It finds loose edges in the network randomly, chooses a source and target road segment and creates a desired amount of routes. The project Osm2po [24], which is able to create routes, is used for this purpose. It uses a weighted graph, created out of OSM XML data and contains multiple routing algorithms based on Dijkstra/AStar. Osm2po can be included very easily by just one Java library, and is therefore also platform independent and does not need any extra installation. TraffSim also supports dynamic routing, which is explained in detail in chapter IV.

I. User Interface

The UI is built up modularly, using eclipse RCP views for the different displayed content, as figure 9 shows. The main area includes an overview of the current state of the simulation with the road network and all vehicles inside it. It allows zooming and translation to all directions, as known from digital map applications like OSM. Moreover, maps from various providers can be used as background layer, including OSM (fig. 10a), Google Maps or Bing Maps for road display or also satellite or hybrid (10b) layers. This allows users to gain a better orientation for known locations, and also enables validation of the loaded road network.

The lower part of the main window is used for statistic views by default. But the RCP environment allows moving, relocating or even detaching of views as desired, so that users can build up their layout as they like to. All other views (setting up traffic lights, exporting the map, finding items, details view, ...) are opened detached by default, since they are usually not used permanently.

IV. Congestion Detection and Rerouting

With the purpose of changing a vehicles route so that delays can be avoided as good as possible, calculation of a new route which is an alternative to the current one while bypassing the congested road segment(s) is necessary. This is implemented by using the Osm2po library, which is also in use for pre-calculation of the routes.
A. Automatic Congestion Detection and Routing Graph Weighting

The detection of congested road segments as well as the selection of relevant vehicles which should be rerouted due to this very congestion is closely intertwined. TraffSim contains multiple approaches how to take care of this issue.

1) Continuous Weighting: The dynamic graph which is used for calculating the route is updated depending on the current situation on the road. If not occupied by any vehicle, each road segment has a defined weight $w = v_{\text{limit}} \times s_{\text{road}}$, where $s_{\text{segment}}$ is the road length. As soon as any vehicle enters a specific segment, the weight is re-calculated cyclically according to equation 3. Variable $n$ represents the number of vehicles driving along the road segment and $v_i$ is the speed of vehicle $i$.

$$w_{\text{seg}} = \frac{1}{n} \sum_{i=1}^{n} v_i \times s_{\text{segment}} \quad (3)$$

By using this dynamic weight, the routes are updated continuously.

2) Threshold Methods: Another approach is to narrow the set of relevant vehicles by setting a defined threshold. This way it is defined whether or not a specific situation is treated as congestion, including necessity to reroute identified vehicles. A simple approach is to apply the threshold using occupation of the road as parameter, as done in [10]. A maximum amount of vehicles per road segment is calculated by dividing the current amount of vehicles of a road segment by the maximum amount and compare with the threshold. Graph weighting is also done by using the relation between current amount of vehicles compared to the maximum amount of vehicles that is possible on the current road segment. This amount is calculated by taking into account the average vehicle length, minimum gap between vehicles, the road length and number of lanes of the road segment.

3) Forecast of Situation on the Roads: In addition, we developed a threshold approach which applies calculations of classifications for each node and time interval. A node is basically a road crossing with at least three input road segments. The number of vehicles passing this node within a particular time interval is calculated by using the forecast method, which is explained below. The observation interval can be defined arbitrarily, same is true for the threshold. The weighting of the graph can be done by one of the two methods presented above.

B. Vehicle Selection

1) Unrestrictive: Certainly, the most straightforward approach is to simply carry out no selection of vehicles, but always calculate new routes for all vehicles in the system at the expense of performance. Furthermore, it may not always be the most effective solution, since vehicles which are very far away from a congested segment and maybe would not be influenced by a short-term jam.

2) Selection by Range: As soon as a congestion emerges, vehicle which should be considered for rerouting are selected by a defined range. This can be measured in a number of road segments [10] or a circuit around the traffic disruption calculated using distance or time. This method has the disadvantage that first vehicles which are not affected by the disturbance could also be selected for rerouting, and second, vehicles which are outside this circle are not considered at all.

3) Selection of Vehicles Passing the Congestion: The third approach takes only vehicles into account whose routes go past the reported congestion. As a result, rerouting is possible over a wider area while keeping unnecessary route calculations minimal. An extension to this approach is the prediction of travel times until the vehicle reaches a certain node of disturbance. Certainly, this can only be applied if not only the disturbed location is delivered by the congestion detection algorithm, but also the prognosticated time. For this to work, TraffSim maintains three-dimensional matrices which contain mappings from nodes to estimated timestamps and vehicles that pass the particular node. The time is estimated by using the dynamic routing graph, which further contains time estimations for each segment to be passed.
C. Dynamic Rerouting

1) Standard Shortest Path: As soon as a part of the road network shows signs of congestion and a set of affected vehicles is determined, a new route is calculated for those. The rerouter uses the weighted graph, which is update with one of the methods described in section IV-A. Based on that, the route with the shortest cost is determined using the osm2po library from [24].

2) K-Shortest Path Method: This routing method calculates a number of $k$ shortest paths according to [26]. This method is also used in [10] and it tries to minimize relocation of the detected traffic jam to the alternative route. As selection method which of the $k$ calculated paths is used, we use another weighting method which counts the usage frequency of a single path by looking at other vehicles’ routes. The algorithm selects the path with the least usage rate from other vehicles and by that very fact tries to find the optimal route.

V. SIMULATION APPLICATIONS

TrafficSim can be used for various situations where microscopic traffic simulation is necessary and useful. Any research based on vehicular traffic using travel times, fuel consumption or average speed is applicable and relatively easy to configure within TraffSim. Simulations for multi-lane highways including entrance and exit ramps have been already executed, as well as for dense road networks within cities. Within this research project, TraffSim will be used to simulate real environments and find out how travel time, fuel consumption and travel distance change if vehicles get information very early and bypass congested road sections.

VI. FUTURE WORK

Although the current version is stable and able to create utilizable results, TraffSim is still under development. The following enhancements and extension have been thought of or are planned in the near future:

A. Consideration of Terrain Geometry

Hills and slopes have considerable impact on fuel consumption, as indicated in [27]. For a more precise calculation of fuel consumption, it is therefore suggestive to use altitudes in the imported map and include them in the calculation. This data could for example be mapped from Shuttle Radar Topography Mission (SRTM) from [28], which is available for free in a resolution of about 90 meters almost all over the world.

B. Intelligent Route Creation

The source and target of vehicles should be found via intelligent heuristics rather than randomly. For example, the population density or traffic measurements can be used to identify the routes.

C. Definition of Multiple Target Nodes

Due to resource limitations, simulations can hardly reproduce the road network of the whole world. Instead, in most cases an area of interest is cut out so that statements valid for this specific area can be extracted from simulation results. We think of situations with lots of commuters trying to move from their home in the surroundings of a bigger city to their workplace in the morning and back home in the evening. It may not matter for them which entrance or exit road of the city they use, assuming there exist multiple roads with comparable cost of time. For simplification of those simulations, the area could be selected so that only the center of a city is modeled, while excluding the surroundings and just setting one of the exit roads as target nodes of the routes. Same is possible vice versa, by excluding the city center, model the surroundings and set the entrance roads as targets.

VII. CONCLUSION

Simulators are very important components of intelligent traffic management systems. Design and implementation of them is a challenging problem, which is solved by this paper. We present TraffSim, a microscopic, parallel and time-continuous traffic model for simulating real traffic in real environments. It allows easy comparison of input parameter changes by parallel simulations and generates fully reproducible results, while also making use of multithreading for achieving good simulation performance. Its architecture is designed modular to encapsulate the functionality and provide mechanisms for extension and adaption of features. Implementations of car-following models for longitudinal movement, a fuel-consumption model, a lane-change model as well as intersection controls for unregulated and traffic-light controlled intersections are built in. A concurrent crash detection ensures safe and realistic behavior. The statistics component records appropriate data for vehicles, road segments and intersections for result analysis in postprocessing. The user can control and observer simulations through a representative Eclipse RCP user interface. For investigations concerning optimization of driving times, distance or fuel consumption, a built-in rerouting framework with multiple configuration possibilities is integrated within TraffSim. This includes proposals for congestion detection, selection of vehicles which are to be considered for rerouting as well as dynamic calculation of alternative routes to the original target node based on the current traffic situation.

ACKNOWLEDGMENT

This project was supported by the program Regionale Wettbewerbsfähigkeit OÖ 2010-2013, which is financed by the European Regional Development Fund and the Government of Upper Austria.
C. BACKFRIEDER et al: TRAFFSIM - A TRAFFIC SIMULATOR FOR INVESTIGATIONS OF CONGESTION ...

REFERENCES


