From GIS to Mixed Traffic Simulation in Urban Scenarios

Jörg Dallmeyer, Andreas D. Lattner
Information Systems and Simulation, Goethe University Frankfurt, POB 111932
60325 Frankfurt am Main, Germany
{dallmeyer, lattner}@cs.uni-frankfurt.de

Ingo J. Timm
Business Informatics I, University of Trier
54296 Trier
ingo.timm@uni-trier.de

ABSTRACT

Simulations are widely used for modeling, analysis, planning, and optimisation of traffic flows and phenomena. For realistic traffic simulations within urban scenarios, the following tasks have to be solved: (1) modeling of the road structure; (2) specification of the behaviour on the road. In our days, very detailed road models for almost any major city exist in Geographic Information Systems (GIS). In the last two decades, the Nagel-Schreckenberg model (NaSch) has been established as de facto standard for car behaviour in freeway traffic due to its efficient and realistic simulations. Within urban scenarios, NaSch lacks of flexibility to integrate heterogeneous road users like cars and bicycles.
The tasks mentioned before are addressed in this paper, i.e., we propose an approach for modeling and specification of urban mixed traffic simulations. As a first step (1), an extended graph as basis for traffic simulation has to be designed. For a concrete scenario, it will be automatically generated on basis of *OpenStreetmap* cartographical material. The specification of road user behaviour (2) has been influenced by the NaSch model. However, the model has been extended to cover the lack of NaSch in urban scenarios: A non cell-based approach is chosen for traffic movement. Furthermore, the routing of traffic users is based on either probability or A* based routing. In this paper, details on the modeling and specification are presented and experimental results are provided.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Traffic Simulation, Urban Scenarios, GIS

1. INTRODUCTION

With a continuous growing amount of road users, the number of traffic jams, transportation costs and gasoline consumption increases. The field of geological inspired traffic simulations has therefore become important in science and industry. A wide range of publications is available, dealing with traffic simulations for the prediction of certain events, like traffic jams. It is important to have a powerful simulation tool, in order to be able to forecast such events on time and to take measures in order to have an impact on the events. Another fundamental application for traffic simulations is the optimisation of traffic lights for a city network, an area with much optimisation potential.
The field of traffic research is an old-established field of research, which has started with makroscopic models, describing the flow of traffic with help of fluid dynamics models. Microscopic models, describing a simplified behaviour of road users to allow for emergent effects, e.g. traffic jams, have been established later. The defacto standard for this field, the Nagel-Schreckenberg model, is efficient to compute as it is based on a cellular automaton model. The model is able to reproduce observed effects on freeway traffic.
The work presented here modifies the model to be adequate for inner city *and* freeway traffic. In addition, the modification allows to simulate traffic with different kinds of road users.
The presented simulation tool is intended to simulate the traffic road networks of real world cities and their surrounding areas. Since the *OpenStreetmap*[1] initiative gives the opportunity to work with cartographical material, updated on a daily base, we decided to integrate this material in our simulation system. The simulation is therefore built on a Geographic Information System (GIS), which is based on the free toolkit *GeoTools*[2]. An underlying GIS gives the freedom to import environment models from heterogeneous data sources and to combine them in one simulation.
This work is structured as follows. In section 2, a brief survey of related work is given. The following section 3 describes the way from imported GIS layers to a graph datastructure for simulation. Section 4 discusses the used model for traffic behaviour, followed by a briefing on the used simulation control in section 5. The evaluation results are presented in section 6. The article closes with a conclusion and an outline of the future work in section 7.

2. RELATED WORK

This work is governed by work of the field of Geographic Information Systems (GIS) and GIS based simulations. The

SIMUTOOLS 2011, March 21-25, Barcelona, Spain
Copyright © 2011 ICST 978-1-936968-00-8
DOI 10.4108/icst.simutools.2011.245680

[1]http://www.openstreetmap.org
[2]http://www.geotools.org/

benefit from the usage of a GIS for simulation scenarios was shown in [28, 27]. The appliance for traffic simulations was done in [13] for the prediction of traffic air pollution in urban scenarios. However, a deterministic model without regarding the impact of emergent effects in traffic was used. Nagel and Schreckenberg invented the Nagel-Schreckenberg car behaviour model [20] for freeway traffic. The model was extended for multi lane traffic in [21, 15] and compared to real world traffic observances [31]. Since the model was intended to simulate freeway traffic, adaptations were made to fit it into urban traffic scenarios [10, 12]. Barrett et al. discuss the idea of building multi resulution cellular automaton (CA) models for traffic simulation [2] on the cost of computation time efficiency. The method of scaling the cell size of the CA model down to an infinitesimal size is discussed in [16] but without the integral idea to use this method for the simulation of heterogeneous road users like cars, trucks and bicycles. The presented approach is still cell-based and not actor-oriented.
A popular simulation system is the *TRANSIMS* project, enabling to simulate multi-modal traffic scenarios [18]. Much work on this simulation was published in the past years even for urban scenarios [19] with the focus on how agents plan their days and routes through the city network. Our work tackles the interaction between different kinds of road users. Traffic light optimisations were presented, working on synthetical graphs [4, 26]. The optimisation of traffic lights in these works is only done in synthetical scenarios and not on the road maps of real cities. The following sections 3 to 5 discuss how the implemented simulation tool is designed and what the similarities and differences to the presented works of this section are.

3. BUILDING AN EXTENDED GRAPH

In order to accomplish a traffic simulation on real world cartographical material, a graph data structure has to be generated out of an imported road map. This section briefly describes the process of the generation of the data structure *ExtendedGraph*, which is the fundament of the simulation system. It is basically a graph data structure with undirected edges as described in [7], but with the possibility to store more information in the nodes and edges. Nodes are represented in a data structure, called *NodeInformation* (NI), edges in *EdgeInformations* (EI). Each NI is connected to at least one EI and each EI is connected to exactly two different NIs: NI_A and NI_B.
The input for the graph generation algorithm is a shapefile, containing features[3] representing roads with different attributes (for instance “Id”, “Name”, “Bridge”, “Maxspeed”, “Oneway”, “OSM_id”, “Ref”, “Type”, “Geometry”)[4]. All features of an user defined area are put into an instance of the *GeoTools* `FeatureGraphGenerator`, which produces a graph. All edges and nodes of this graph are converted into EIs and NIs, afterwards. This step provides the opportunity to store additional information in the graph. All EIs are stored in a collection, called *EdgeInformationCollection* (EIC). NIs get stored in a *NodeInformationCollection* (NIC).
The graph needs some processing steps, to convert it into a useful graph. The first problem is that roads crossing other roads are not automatically splitted at the point of the intersection.
Roundabouts can be represented with only one EI. This results in EIs, being connected to only one NI, which breaks the rule of each EI having exactly two different NIs. This rule is important, because it gives a simple method to know, into which direction a simulated road user is moving: Each road user is moving on a certain EI and has visited a certain NI at last. To solve this problem, EIs with identical NI_A and NI_B are split at the center of their geometry and a new NI is inserted at the splitting position.
In order to reduce the complexity of the graph, useless NIs get removed from the graph. NIs with order 2 are NIs without the functionality of a branch connection and get therefore removed and the corresponding EIs get merged. This is only done if the resulting EI does not match the roundabout criterion.
Features of the read in shapefile may have the attribute “Bridge” set to “1”, which means, that the corresponding road is a bridge. This attribute is disregarded by now. This results in bridges crossing normal roads, but anyhow being connected to them. To solve this problem, NIs connecting two parts of a bridge with two parts of a normal road, are split into two NIs. The first NI for the connection of the bridge parts and the second for the normal road.
Subsequently, additional information is put into the graph. The implemented *ExtendedGraphGenerator* automatically reads out the positions of traffic lights from another shapefile and adds instances of simulation traffic lights to the corresponding NIs. The information how to turn (left, right, ahead), in order to move from one EI_1 over a NI to another EI_2 and the angle between EI_1 and EI_2 is stored in the corresponding NI. This information can be computed out of the stored geometries of the EIs and is important for the determination, which road user is in the right of way at a crossing. The positions of parking spaces are read in from a shapefile for polygon geometries. Each EI, being part of a parking space is marked and put into a collection called *EdgeGroup*. The different parking places are administrated by different *EdgeGroups*.
In order to model the basic features of real world roads, it is important to specify how many traffic lanes an EI for a corresponding road shall have. This information is not given by the road layer. It is therefore assigned in respect to the type of the EI (e.g., “residential”, “secondary”, “footway”, “living_street”) using a lookup table (LUT) (see [3]). An identical procedure is used to determine the maximum velocity for an EI, if this information is not given. At this point, a refinement has to be done, since a typical $v_{\max}$ for the type “secondary”[5] is $100\text{km}\cdot\text{h}^{-1}$, but at some places, secondary roads proceed through a town, where a speed limit of about $50\text{km}\cdot\text{h}^{-1}$ is appropriate. The shapes of all towns, extracted from the polygon layer, are used to determine all roads being located within towns. The $v_{\max}^{\text{EI}}$ for these roads is set to

$$v_{\max}^{\text{EI}} = \min\left(v_{\max}^{\text{EI}}\ ,\ 50\text{km}\cdot\text{h}^{-1}\right) \qquad (1)$$

[3]After [14, p. 62f], a feature is the smallest spatial entity in a Geographic Information System. It normally contains spatial information (e.g., data structures like polygon, linestring or point) and additional information (e.g., the name of the modelled road).

[4]You can find example files from GEOFABRIK on the website http://download.geofabrik.de/

[5]Secondary roads are not part of a major route, but form a link to the national route network. They are therefore usually roads with two lanes.

if the original $v_{\max}^{\text{EI}}$ was determined via usage of the LUT and was not a given information from the read in road map. Major roads are identified via usage of a third LUT, which defines the value of this flag for each type of road. In German towns and cities, the amount of roundabouts increases. Roundabouts normally have a higher right of way than all other road types. Even if a road user wants to enter a roundabout from a major road, the roundabout has the right of way. Therefore, roundabouts have to be identified in the graph. A roundabout has a few representative characteristics. All parts of the roundabout have identical values in respect to name, road type, maximum velocity, number of lanes and roundabouts are always oneway streets. The geometries of the parts of a roundabout form a circle in anticlockwise direction. With the determined characteristics, roundabouts can be found in the graph. The right of way of a road has to be extended to a numeric value, where the higher the value, the higher the right of way. Now, roundabouts can have a higher right of way than major roads, which get a higher right of way than normal roads.

4. TRAFFIC

After the *ExtendedGraph* is built, road users have to be simulated on it. In the field of traffic simulation, different elementary approaches can be used to model the movement of traffic. In macroscopic models, traffic is decribed with help of variables like traffic flow and traffic density. In microscopic models, individual road users get modeled and the macroscopic results arise due to emergence effects. Different car behaviour models have been published in the past. One of the most established models is the Nagel-Schreckenberg model (NaSch) for freeway traffic [20], which can be calibrated to reproduce the results of a linear car-following model and a kinematic wave model [8]. It has been shown, that it can be used to simulate the macroscopic effects of real world highways [31].

4.1 A cell-based approach

Before we describe our approach in section 4.2, a short introduction into the NaSch model has to be given. In NaSch, roads get partioned into cells with a length of 7.5m, which is an estimation for the required space for one vehicle. Each cell may contain a car or may be empty. In each simulation step, each non-empty cell performs algorithm 1. Lines $1-3$

Algorithm 1 NaSch model [20]

```
1: **if** $v \leq v_{max}$ **then**
2:     $v \leftarrow v+1$
3: **end if**
4: **if** $v > dist$ **then**
5:     $v \leftarrow dist$
6: **end if**
7: **if** $random \leq prob$ **then**
8:     $v \leftarrow v-1$
9: **end if**
10: $position \leftarrow position + v$
```

describe the operation of acceleration. The car accelerates, if possible. If the distance *dist* to the preceding car is smaller than the current speed v, the car adjusts its speed to the distance (lines $4-6$). With a certain probability *prob*, the car brakes without a deeper reason (lines $7-9$, where *random* is a random number between 0 and 1). Afterwards, the car moves with the calculated speed (line 10). The functionality of NaSch is shown in figure 1.

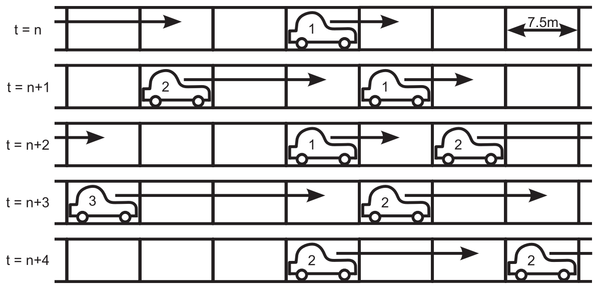

Figure 1: NaSch model

velocity [cells]	0	1	2	3	4	5
velocity [km $\cdot$ h^{-1}]	0	27	54	81	108	135

Table 1: Possible velocities in NaSch

The described model is a cellular automaton model, which is efficent to compute. It was adapted by Rickert et al. to simulate the whole german highway traffic [24]. The NaSch model has turned out to be a minimal model which reproduces the basic emerging effects on motorway networks. A wide range of literature has been published, using or extending the NaSch model (e.g., [1, 17, 32, 4, 11, 12, 6]).

In NaSch, v can only be an integer multiple of the width of one cell. The maximum velocity v_{max} is normally set to 5 (or $v_{max} = 2$ for urban scenarios [10]), so that each car can move with one of the velocities shown in table 1. Thus the model has to be adjusted to fit into the simulation of urban traffic, where the speed steps have to be smaller. Fouladvand et al. change the cell width to 5.6m, which leads to smaller speed steps [12]. Additionally, each time has a duration of two seconds to halve the speed of each car in the simulation. The resulting speed steps are 10km $\cdot$ h^{-1}.

In urban scenarios, different kinds of road users (e.g., fast / slow cars, bicycles, trucks / buses) drive with different velocities, amounts of acceleration/deceleration and safety distances. A first approach to solve this problem is to take the smallest needed cell width for every kind of road user and scale the velocities for each type. This approach has the shortcoming that the whole system has to be rescaled, whenever a new slowest kind of road user shall be implemented. The upper limit of granularity would be reached, when slow pedestrians shall be simulated on sideways. This would result in an intense increasement of the number of cells. Another disadvantage of the cell-based approach for urban scenarios is that the length of innercity roads is not often a multiple of the cell width which results in lossy approximations of a given road map.

4.2 Non cell-based approach

The basic algorithm of car behaviour in our approach is an generalisation of algorithm 1. The implemented simulation skips the discretisation of roads into a sequence of cells. Roads are modeled as edges with lengths in m. Velocities are calculated in m $\cdot$ s^{-1}. This is shown in figure

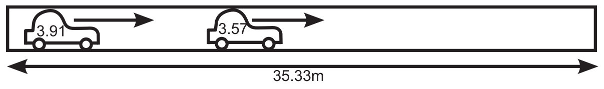

Figure 2: Non cell-based approach with floating velocities

2. Each car has a dally factor τ which determines, how often the car will dally. The value of τ is normally distributed with $\mu = 0.3$ and $\sigma = 0.1$ and limits 0 and 1. Most cell-based models use a fixed dally factor 0.3 [5]. The acceleration $a(v)$ for the current velocity of the car v is calculated with a combination of the τ and the maximum velocity $v_{\max}^{\text{car}}$ of the car, which is uniformly distributed between $v_{\max}^{\text{car}} = \left[100\text{km} \cdot \text{h}^{-1} \cdots 200\text{km} \cdot \text{h}^{-1}\right]$. With these simple methods, varying driving styles can be accomplished. The upper bound of velocity is the minimum of the $v_{\max}^{\text{car}}$ and the maximum velocity prescribed on the edge v_{max}^{EI}. Each car holds a safety distance $sd(v)$ which is linear dependent to v. The used method is shown in algorithm 2. Whenever a car

Algorithm 2 Non cell-based approach

```
1: if $v \leq \min\left(v_{max}^{car}\ ,\ v_{max}^{EI}\right)$ then
2:     $v \leftarrow v + a(v)$
3: end if
4: if $v + sd(v) > dist$ then
5:     $v \leftarrow dist - sd(v)$
6: end if
7: if $random \leq \tau$ then
8:     $v \leftarrow v - a(v)$
9: end if
10: $position \leftarrow position + v$
```

arrives at an intersection, it has to check whether it has the right of way and may pass through or has to wait. At this point, the geometries of the EIs are important for the determinition of the rule "left yields to right"[6]. The real world rules for the right of way are implemented in the simulation model. It is important to consider roundabouts because of the increasing amount of roundabouts in inner cities. Each road user has a public turn indicator variable which makes the turning more realistic. A road user ru_1 thus has not to wait until another road user ru_2 with the right of way passes the way of ru_1, when ru_2 indicates, that it will turn into a direction that does not touch the way of ru_1.
Different types of road users can be modeled by the use of the described approach. Exemplarily, bicylces are modeled in our simulation. A bicycle has a lower v_{max} and needs another acceleration function. It has divergent types of roads it can use, compared to cars. For example, a bicycle can use cycle ways and is not allowed to use motorways. In the routing method, a bicycle prefers to use designated cycle ways prior to normal roads. A bicycle does not need much space to overtake another bicycle, so that it can do so, whenever it wants to. A car has to take a look for the gap to the next oncoming car and the gap in front of the bicycle it wants to overtake in order to have sufficient space to get back on the right lane in front of the bicycle. Bicycles are therefore slowing down the car traffic in a city. At this point, the model for bicycle behaviour gets extended to not slowing down (lines $4 \cdots 6$ of algorithm 2), when the preceding road user is a car which is currently overtaking.
Whenever a road user is put into simulation in an urban area, it waits until a traffic gap of the needed size for speeding up arises and then joins the simulation by starting to drive. When a road user has reached its final destination, it parks by slowing down till standstill.

[6]In Germany, cars use the right side of the road. Whenever two cars try to pass a crossing point from two roads with identical rights of way (e.g., two living streets), the car, whose counterpart comes from the right side, has to wait.

4.3 Routing

Each simulated road user has to determine its way through the extended graph. Rickert and Wagner describe background traffic in [23] with identical behaviour as the routed cars. In their work, cars of the background traffic are put on a road and drive until they pass a terminator. From that point on, they drive in the opposite direction. This approach is sufficient for highway traffic, but not for urban traffic. This section describes an efficent routing method for background traffic and foreground traffic.

4.3.1 Probability based routing

Each *NodeInformation* NI stores an utilisation probability list which assigns a probalitiy $p_t(EI)$ to any $EI \in \Omega_{NI}$ where Ω_{NI} is the set of EIs, which are connected to NI. This value describes the probability that the *EdgeInformation* EI will be taken from the point of NI for a road user of type $t \in$ [car, bicycle].
The best way for setting the probability values to the NIs is via calculating them from the information of a traffic count study for the simulated road network. Apparently, it is very time and money consuming to count the traffic for different times of a day for a whole city and different days of the week. A simple approximation for the utilisation probabilities is to analyse the connected EIs of the corresponding NI and to set the probabilities in relation to the kind of the EIs (e.g., allowed speed, number of lanes, type of the road). A "bigger" (e.g., faster and/or more lanes) road is intended to be more important than a "smaller" road and thus gets a higher utilisation probability. This implies that the road design of the used road map corresponds correctly to the real world road usages. It is obvious that this method is very simple to compute, but not very exact. It is therefore used to determine the utilisation probabilities for roads outside the main simulation area. This area is important for flow in and out of the main area Θ.
The probabilities for the main area Θ are computed by determining the shortest ways from each NI_a to each NI_b, where ($a \neq b$ and NI_a, $\text{NI}_b \in \Theta$) and summing up, how often each EI is used from the connected NIs on the calculated ways. The more often one EI is used, the more important it is for the simulated road users and thus the higher the utilisation probability is. The described method has to be applied twice: Once for cars and once for bicycles, because different road user types use different routes. The calculation may take a while and the results are therefore stored in files. A XML-format is used to describe the data. This approach leads to a routing, which results from one analysis step, hence is very efficient during the simulation.

The following equation holds for each $NI \in NIC$:

$$\sum_{EI \in \Omega_{NI}} p_t(EI) = 1 \tag{2}$$

When a road user of type t computes which EI_2 to use when visiting NI from EI_1, the probability $p_t(EI_1)$ is set to $p_t(EI_1) = 0$. This guarantees that no road user drives the road back where it came from, instantly (no u-turning). Afterwards a random number $r \in [0 \cdots 1]$ is calculated and scaled with the factor f:

$$f = \sum_{EI \in \{\Omega_{NI} \setminus EI_1\}} p_t(EI) \tag{3}$$

$$\psi = r \cdot f \tag{4}$$

EIs which are only accessible with an angle bigger than α or EIs which are oneway streets in the wrong direction are sorted out on the same procedure. The used value is $\alpha = 110°$. After this step, a while loop iterates through the list of EIs of the NI and sums up the utilisation probabilities of them until the sum exceeds the value ψ. The last touched EI in the while loop is the resulting EI_2.
Most of the traffic routing can be modeled by the two described methods. It is important to have the possibility to define another group of road users: Those who calculate their way from a source NI to a destination NI. This group does not move randomly and can be used as a control group which drives surrounded by traffic which behaves realistic but whose routing is not very computational expensive. It is not possible to store each shortest path efficiently during the simulation, because the memory requirements grows quadratic to the number of NIs in the *ExtendedGraph*. The control group uses a variation of the A* search algorithm [25]. The weight of an EI between two NIs is defined as the length of the geometry of the connecting EI scaled with the allowed speed for the EI. The lower the weight of EI, the shorter it will take to use EI. An incentive function determines a bonus value for EI in relation to the roadtype of EI and the type of the road user who wants to use EI. For example, a bike path will get a low weight for a bicycle, because it is distinguished to be better to use designated ways for bicycles instead of using ways with car-traffic. The A* search algorithm is used for calculation of the utilisation probabilities as described above and during the simulation for the small control group.

4.3.2 A* based routing

The A* search algorithm has a worst case time complexity of $O(|E| \cdot lg(|V|))$[7], but is faster in the average case. By limiting the area of the graph, which is used to calculate the A* search algorithm, the complexity can be reduced. Nevertheless, it takes much more time to calculate a way via usage of A* search than by the simpler methods, described above. Thus the group of road users, which determines its way by A* search, has to be small. In the simulation, a parameter can be set to define the amount of road users of this group, which is currently set to 1% of all road users.

[7]$|E|$ is the amount of edges (size of EIC) and $|V|$ the size of the NIC. After [7], the Dijkstra algorithm has a complexity from $O((|V| + |E|) \cdot lg(|V|))$ in general and $O(|E| \cdot lg(|V|))$ if all vertices are reachable from the source. A* has therefore the same worst case complexity, because it degenerates into a Dijkstra in the worst case.

A*-routed road users determine a destination NI and calculate their way to this point. The other groups are not able to have a predefined destination. Normally, a town can be divided into different areas. Some areas are mainly used as residential areas and other districts are used as commercial areas. A traffic flow from residential areas to commercial areas and vice versa can be ovserved during rush hour traffic. The simulation of the whole routine of the day is important for traffic simulation. Therefore, a method was developed to steer the probability-based routed simulated road users.
At first, one NI has to be defnined, which is the destination NI_{dest}. A simple approach to determine the way to NI_{dest} from another NI is to choose the $EI \in \Omega_{NI}$ with the lowest Euclidean distance to NI_{dest} for routing with a certain probability. The strength of this approach is its simplicity of computation. On the other hand it has the weakness, that the Euclidean nearest NI it not necessarily the NI with the lowest distance, a road user has to cover on the way to NI_{dest}. The method leads to road users transfixing in a part of the map and not getting their way to their destination NI_{dest}.
In a more realistic approach, the utilisation probabilities of each NI get adjusted in a way, that it is more likely to use the EI which is on the *shortest way* to NI_{dest}. From each $NI \in \{NIC \setminus NI_{\text{dest}}\}$, the way to NI_{dest} gets calculated via A* search algorithm. The utilisation probabilities for each NI get adjusted by adding a bonus value for the EI which is the first EI on the way from the corresponding NI to NI_{dest}. The result of this method is written in a XML document. The simple method leads to a traffic flow in the direction of NI_{dest} without a relevant loss of computation time. The enlargement of this method for a group of destination points can be simply achieved by calculating the method for more than one NI_{dest}. A speed up of the algorithm can be achieved via calculating the way from one NI to NI_{dest} and giving the bonus to each EI on the calculated way and removing each NI on the way from the list of NIs which have to be analysed by the algorithm.

4.4 Traffic lights

As described in section 3, the positions of traffic lights are read out of a shapefile. The concerning NIs get equipped with instances of traffic lights. The road users query a NI for the existence of a traffic light, before passing it. If a traffic light exists and it shows “red”, the NI is not passable and the road user has to wait. The following road users wait with the described safety distance because of line 5 in algorithm 2.
Different types of traffic lights can be modeled. Presently, traffic lights switching by the rule of the highest load are implemented as well as pedestrian lights. The load of a road can be computed in a simple way, because the concerning EI manages the road users driving on it. In reality, this could be done with two ground contact sensors, one at each end of the road. The pedestrian lights are currently having a lack of pedestrians in the simulation. Thus the simulation control simulates a hit of the button of these traffic lights, occasionally. Additionally, the very simple method of switching green in strict rotation is implemented.

4.5 Multi lane traffic

The simulation is not limited to inner city traffic. The inflow and outflow of city traffic is important for the allocation of road users. Another point is the simulation over the geographical range of a few towns, where the interchange of traffic is important. Thus it is necessary to include the opportunity to simulate multi lane traffic. Extensions of the NaSch model for multi lane traffic were described in [23, 21, 15, 32]. They can be simply adopted for the invented non cell-based approach. A road user ru changes its lane to the left if there is a lane on the left and the road user in front of ru is slower than ru wants to drive. Then ru has to check, whether there is a sufficient gap on the left side. The switch back to the right lane is done when it is no longer benefiting to stay on the left, because ru could drive on the right without loss of speed and there is an adequate gap. This approach is sufficient for roads with any number of lanes, but has to be extended by the rule that lane changing to the left is only possible when the iteration number is even and to the right if the iteration number is odd. This guarantees that on a road with three lanes, collisions caused by one road user changing its lane from the right to the middle lane and another road user simultaneously changing from the left to the middle lane, are avoided.

5. SIMULATION CONTROL

The simulation control has the function to steer the simulation. One basic control class provides the needed procedures to build subclasses for different experiments. The simulation control regulates the number of simulated road users, the percentaged distribution between cars and bicycles and the distribution of the two basic routing technologies (A* based, utilisation probabilities). Stimuli, like an increasement of the number of cars can be achieved, easily. The basic main loop of simulation control is shown in algorithm 3. The

Algorithm 3 Simulation control, main loop

```
1: while $condition$ do
2:    $iteration \leftarrow iteration + 1$
3:    $EIC.sort()$
4:    for all roadUser $ru$ do
5:       $ru.update(iteration)$
6:       if $ru.ended$ then
7:          $remove(ru)$
8:       end if
9:    end for
10:   for all roadUser $ru$ do
11:      $ru.provide()$
12:   end for
13:   $addNewRoadUsers()$
14:   $NIC.updateLights()$
15: end while
```

termination condition *condition* may be everything (e.g. a number of iterations or an under-run traffic flow). In line 3, all EIs of the EIC get sorted in respect of the positions of the managed road users of them. The algorithm *Bubblesort* is used for this job. Because of the low relocationing from iteration to iteration on an EI, the sorting takes normally 1 or 2 loops through the road users of each EI. This step is important to have an efficient method for determining the road user in front and in back of each road user, even if an EI manages a huge amount of road users. The update of all road users is done in lines $4 \cdots 9$. In this loop, every road user calculates the actions to perform in this iteration

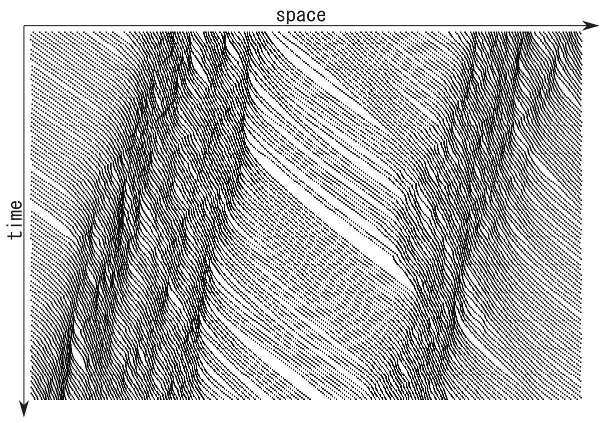

Figure 3: Space Time Diagram

but does not perform them. This has to be done in another loop, afterwards to achieve a synchronous update of all road users. Road users with the sign *ended* are removed from the simulation. A road user sets this flag, when it has driven its calculated way through the graph. New road users are set up with respect to the ratio of the number of currently simulated road users in as-is state and in as-should state. New road users are placed at parking places, EIs at the graph boundary or random chosen EIs, in respect to defined probabilities for each group. The routing method used for a new road user is determined on this way, too. Since all traffic lights are stored in NIs, the NIC receives the command to update the traffic lights. This command is forwarded, till the update-function of each traffic light is called.
The simulation control provides statistical export methods like space time diagrams, vectors of velocities (direct or middled for all road users or all road users of one type). All exports are done with help from *R-project*[8], a free statistical math software. The format of the exports is *R*-code. This is useful to rapidly compare different setups.

6. EXPERIMENTS

This section discusses experiments to investigate the plausibility of the non cell-based approach and its routing methods. At first, the basic functionality and traffic effects have to be compared to the original NaSch model.

6.1 Comparison to the NaSch model

A synthetical *ExtendedGraph* without branch connections was built, that models one big circle. The road has a length of 60,000m with only one lane. The maximum velocity for this road is set to $200\text{km} \cdot \text{h}^{-1}$. An amount of 800 cars and 0 bicylces is put onto the graph moving in a circle. In a first simulation, a space time diagram was computed, which shows a small section of the road. In figure 3, each dot symbolises a car being allocated at a position at a given time.
Free flow traffic areas are the areas showing a linear movement. All waves in the diagram represent disruptions of the traffic flow. These are traffic congestions without an external influance[9] being the result of dallying. The tail end of

[8]http://www.r-project.org
[9]In German the so called "Stau aus dem Nichts" (traffic jam

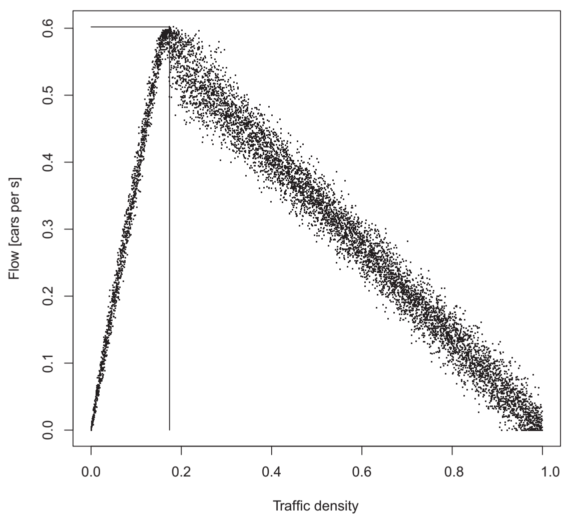

Figure 4: Single lane traffic flow-density diagram

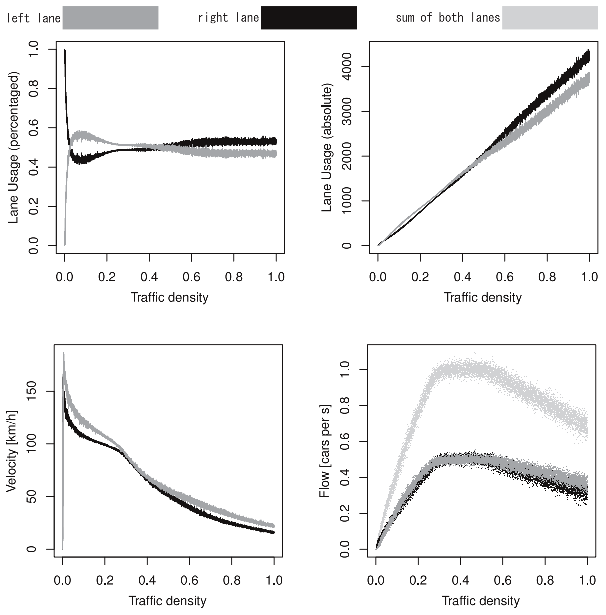

Figure 5: Experimental results for two-lane traffic

the traffic jam moves backwards, which results from new cars arriving at the traffic jam while the cause of the jam does no longer exist. These effects correlate to the described results of [20].

A second test was the generation of the flow-density diagram, shown in figure 4. One ground contact sensor registers, whenever a car passes the sensor. This is implemented in a NI. The systematic generation of the diagram was done as follows: At first, the simulation was run with $i = 0$ cars. The simulation has a settlement phase of 1000 iterations. Then the ground contact sensor gets switched on and the simulation runs for further 1000 iterations. The count of the sensor gets recorded. The simulation gets reseted and $i = i + 1$ cars are simulated and so on.

The minimum needed space per car is 7.5m, thus no car is able to move when $60,000\text{m} \cdot \left(7.5\frac{\text{m}}{\text{car}}\right)^{-1} = 8,000$ cars are simulated on the road. The traffic density is defined as $density = \#cars \cdot 8000^{-1}$. The calculated flow-density of traffic flow matches the results of the literature. The traffic flow in the simulated network is about the doubled flow of the original NaSch-model [20]. This result matches real world observances, as discussed by [9]. The maximum flow is gained at a density of about 0.174 (1391 cars).

The EIs of the synthetic graph now get a second lane and the experiment is repeated with an identical configuration. Additional information is logged from the simulation runs. The resulting diagrams are shown in figure 5.

The comparison to [21] shows an approximately equal flow once again, which confirms the result of the non cell-based method. The maximum flow is gained with a density of about 0.47 (3745 cars). The effects from [15, 30] could be reproduced as follows. The upper part of the figure shows the correlation between traffic density and lane usage. At very low density values, the right lane is suitible for driving, because there are not many slow cars hindering the faster ones. This changes with increasing density. At the maximum flow, a second change of lane usage behaviour can be observed. An explanation for this is the keeping of safety distances. The right lane contains most slow vehicles which results in smaller safety distances on the right lane and therefore a higher capacity for this lane. Thus, the left lane is the faster lane, as shown in the bottom left part of figure 5. The plot is calculated as follows. The average speed of all vehicles per lane is noted after each iteration. At the end of each simulation run for a certain traffic density, the average speed per lane for the whole run is calculated. The bottom right part of the figure shows the flow-density diagram of the two lane traffic experiment. The maximum flow is held over an interval of traffic density. This results in the feasibility to overtake slow cars, what was not possible in the single-lane experiment. The model can be adapted to given road maps, now.

without external influence).

6.2 Read in road maps

At first, a card section of the town "Erlensee" and a small area around the town, including two other towns, was read out of a shape file for the federal state of "Hessen". The *ExtendedGraph* (3,583 EIs; 2,708 NIs; overall road length 637km) was generated. The utilisation probabilities got computed as described in section 4.3.1, where the probabilities for the main area are read in from a XML file. 200 road users are put on the graph with 10% bicycles and 90% cars. In order to simulate events like rush hour traffic or the end of a football game, it is useful to steer the probability based routed road users. The upcoming test runs without A* based routed road users and documents the functionality of the method, described in section 4.3.2. After a settlement phase of 1,000 iterations, the mean distances of all road users to a predefined NI_{dest} are stored for each iter-

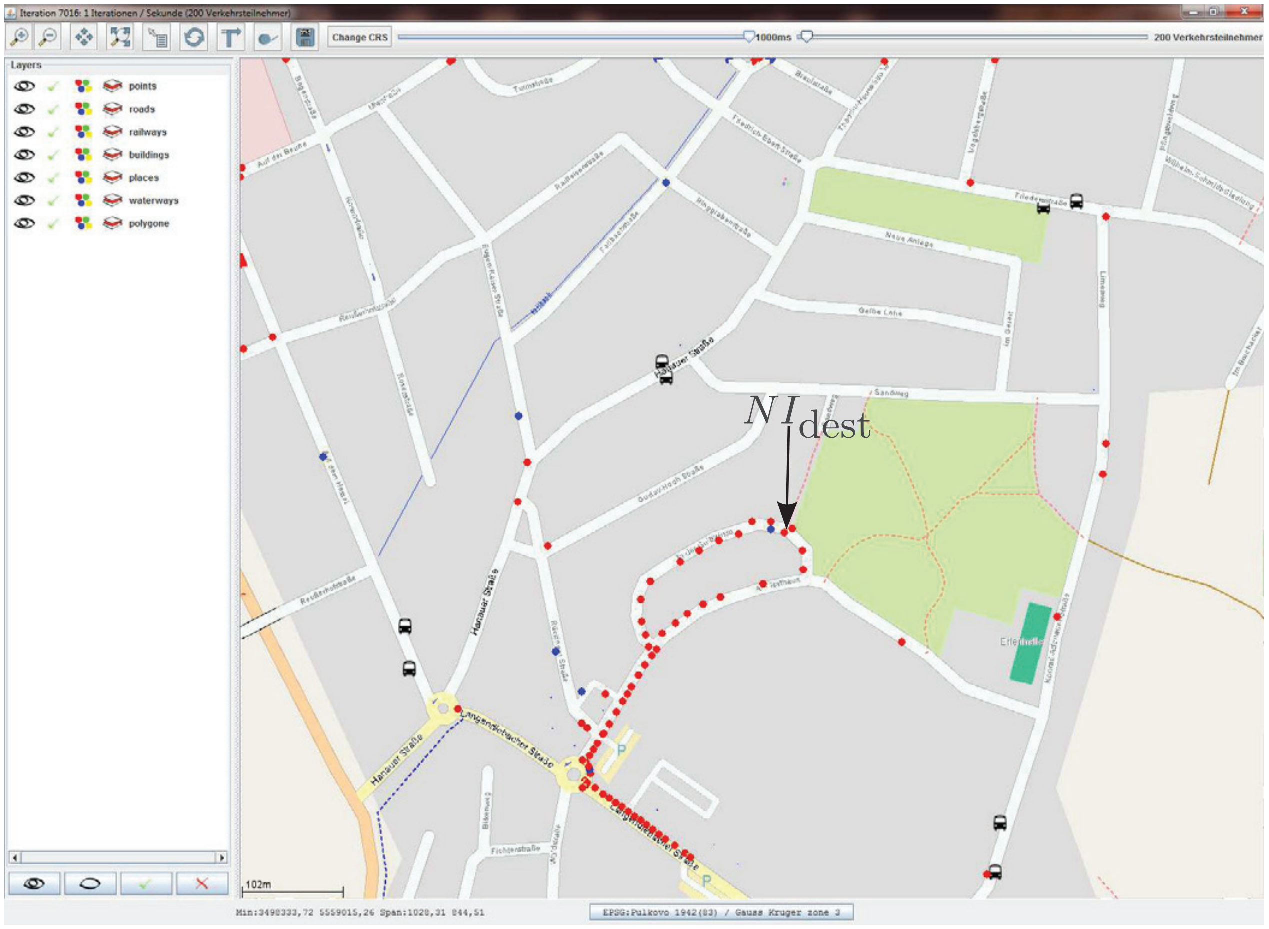

Figure 7: Screenshot taken during experiment. Red bullet: car, Blue bullet: bicycle.

ation. After further 5,000 iterations, bonus values for the utilisation probabilites are added to each NI to direct the road users to NI_{dest}. The distances to NI_{dest} decrease, as shown in figure 6.

The vertical line highlights the iteration in which the utilisation probabilities are modulated. The distances converge to a value of approximately 325m, because of new road users being created at random places during the simulation and the still existent probabilities to use EIs, not being directed to NI_{dest}. Furthermore, the middled distance cannot become 0 because of the space, each road user needs on the roads. This fact is shown in figure 7. It is recognisable, that each road user holds its safety distance to the preceding one. An application for the shown method is the simulation of rush-hour traffic.

The simulation was successfuly tested on the middle sized city Hanau and surrounding towns (11,590 EIs; 8,339 NIs; overall road length 1,479km) and on the major city Frankfurt am Main (24,033 EIs; 16,434 NIs; overall road length 2,048km). The standard amount of road users is 15% of the number of EIs. The simulation of Frankfurt am Main is done with about 150-fold real time on a standard desktop PC (2.66Ghz Intel Core 2 Duo CPU, 4GB ram).

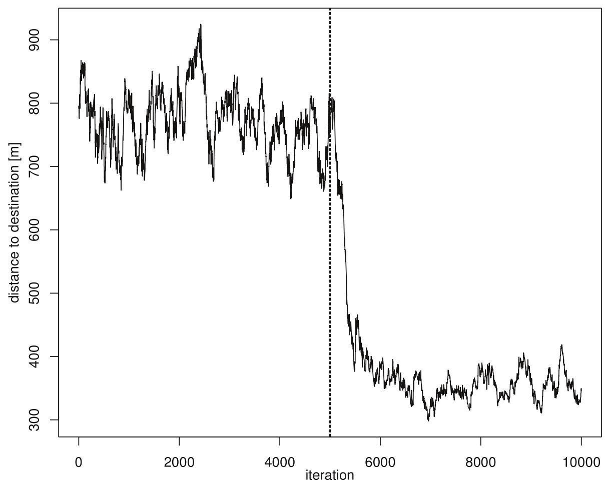

Figure 6: Distance to destination point

7. CONCLUSION & FUTURE WORK

In this paper we have introduced an approach for building a traffic simulation model on the basis of GIS layers. Therefore, we have developed behaviour of the simulated road users as well as different routing methods for urban traffic simulation. We have shown the feasibility of our approach in multiple simulation experiments. As a result, well-known dynamics, e.g., emerging traffic jams, could be reproduced, i.e., realistic traffic simulation is feasible. The use of up-to-date road maps from *OpenStreetmap* allows for traffic simulation of a wide range of cities without extensive effort as the transformation procedure is automated. The model introduced here enables for mixed traffic simulation in contrast to the NaSch approach. The provided export methods make it easy to compare the results of different simulation runs and to build typical diagrams, automatically.

7.1 Work in Progress

The system is currently in a working state. A few improvements and extensions will be integrated in the future. With the implemented methods, the simulation of a whole day city traffic, including two times rush hour traffic, will be simulated in the near future via appliance of the utilisation probabilities, described in section 4.3.1. It is planned to apply machine learning in order to predict events like traffic jams. After this step, the prevention of such events could be implemented via rerouting traffic with help of a simulated traffic management system. The simulated traffic lights need to be optimised. At this point, we believe in a *butterfly effect for city traffic*. We suppose, that the optimisation of a few traffic lights building a green wave is not the best strategy for optimising traffic flow in big cities because of the huge amount of interdependent traffic entities (e.g., road users and traffic lights). It seems to be a better approach to optimise the whole city. This could be done with help of smaller clusters of traffic lights, organising each other with help of information, received from neighboured clusters.

At traffic lights, different kinds of road users behave in different ways. As observed while waiting at traffic lights, especially bicyclists tend to get ahead of the whole waiting queue. This appeares to slow down the whole traffic after the traffic light switched to green, because every car has to reovertake the corresponding bicycles, what is not generally possible because of opposing traffic. First examinations on a synthetic generated road graph exhibit a significantly decreasing traffic flow when cyclists shove at traffic lights. The model of car behaviour has to be refined to simulate this scenario, in detail. Then, it has to be tested on real city maps, to check for the influence for the total traffic flow.

An easy to achieve extension of the model will be the simulation of accidents. One example is the overtaking manoeuvre: A probability could be calculated with respect to the distance to opposing traffic, that an accident may happen with the opposing road user. Then, both road users could easily set their speed to zero and stay on the road. Another possibility is an accident with the overtaken road user, which could lead to the blockade of only one direction on the corresponding road.

Now the question arises, what to do when stuck in traffic. The road users need to be able to *replan*, when the traffic gets stuck. This is especially important for the control group which uses the routing mechanism of A*. A road user ru could replan by choosing a NI in the surrounding area of ru and calculating its way to NI. Afterwards, the way from NI to the original destination of ru can be calculated. In that way, ru could be able to drive around a traffic jam. A route deviation from a NI for the probability based routed road users could be established by varying the utilisation probabilities for all $EI \in \Omega_{NI}$.

In section 4.3.1, the probability based routing was described. The probabilites are read out from a XML file. At this step, the results of calculated ways could be read out, too. If each NI would store a list of destinations, holding the first NI_1 on the way to each NI_{dest}, the path finding in the calculated main area would take $\mathcal{O}(n)$, where n is the number of EIs on the path. Nevertheless it will not be sufficient to let each road user determine its way via A* search, because of the loss of variations. In reality, each road user routes its way with respect to the parameters “quickness” and “personal fancyness of the way”. This is covered more efficiently by the probability based routing method. The load of each road in the simulation has to be compared to real world observances from traffic countings.

In residential areas, parking cars build hindrances on the roads. This fact is currently not implemented in the simulation system, but can be simply done by putting cars with a fix velocity of $v = 0$ in these areas. The overtake models of the road users have to be extended to being able to overtake standing cars, marked als *parking*.

7.2 Perspectives

There is another important group of road users: pedestrians. Very simple and computationally efficient CA models for pedestrian movement have been published [33, 29]. To the best of our knowledge, no one has ever combined these models with a graph based traffic simulation. This step is possible because of the usage of the underlying GIS. One of the next challenges is to put pedestrians on the map, who could use zebra crossings and traffic lights or cross roads without usage of pedestrian crossings. Pedestrians could walk through parks without regard to the marked paths using polygon layers of the GIS. Thus, this would be a combination of very heterogeneous entities in one simulation.

We are holding a digital terrain model (DTM) for the simulation area, received from the “Hessisches Landesamt für Bodenmanagement und Geoinformation”, partitioning the landscape in squares with a side length of 10m. For each defined square, the DTM defines the height of the square. Our simulation tool comes with a parser for the DTM format, converting the DTM file to a shape file for GIS usage and automatically generation the Styled Layer Descriptor (SLD) file, clustering the heights into user defined steps. With help of a DGM layer, the routing of bicycles, pedestrians and motor trucks can be adjusted to be more energy efficient or ridable at all. A more detailed pollutant emission model could be established. Energy efficient routing and traffic light preemptioning are nowadays important and in vogue. Work in this field has to be done. The implemented routing methods should be compared to the iterative routing method, used in *TRANSIMS* [22]. Additionally, we are considering the effects of making the described simulation tool open source.

Acknowledgement

This work was made possible by the *MainCampus* scholarship of the *Stiftung Polytechnische Gesellschaft Frankfurt am Main*.

8. REFERENCES

[1] R. Barlovic, L. Santen, A. Schadschneider, and M. Schreckenberg. Metastable States in CA models for Traffic Flow. *ArXiv Condensed Matter e-prints*, Nov. 1997.

[2] C. L. Barrett, S. Eubank, K. Nagel, S. Rasmussen, J. Riordan, and M. Wolinsky. Issues in the representation of traffic using multi-resolution cellular automata. Technical report, Los Alamos National Laboratory Transims Report Series, 1998.

[3] N. Bartelme. *Geoinformatik: Modelle, Strukturen, Funktionen*. Springer, Berlin, 4. edition, 2005.

[4] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E*, 64, 2001.

[5] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger. *Modellbildung und Simulation*. eXamen.press. Springer Verlag, 1 edition, 2009. ISBN 978-3-540798095.

[6] D. Chowdhury and A. Schadschneider. Self-organization of traffic jams in cities: Effects of stochastic dynamics and signal periods. *Phys. Rev. E*, 59(2):R1311–R1314, Feb 1999.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition, September 2001.

[8] C. F. Daganzo. In traffic flow, cellular automata = kinematic waves. *Transportation Research Part B: Methodological*, 40(5):396 – 403, 2006.

[9] H. Emmerich and E. Rank. An improved cellular automaton model for traffic flow simulation. *Physica A: Statistical and Theoretical Physics*, 234(3-4):676 – 686, 1997.

[10] J. Esser and M. Schreckenberg. Microscopic simulation of urban traffic based on cellular automata. In *International Journal of Modern Physics C*, volume 8, pages 1025–1036. World Scientific Publishing Company, 1997.

[11] M. E. Fouladvand and N. H. Radja. Optimised traffic flow at a single urban crossroads: Dynamical symmetry breaking. Technical Report cond-mat/0108149, Institute for Studies in Theoretical Physics and Mathematics ,P.O. Box 19395-5531, Tehran, Iran, Aug 2001.

[12] M. E. Fouladvand, M. R. Shaebani, and Z. Sadjadi. Intelligent controlling simulation of traffic flow in a small city network. (physics/0511141), Nov 2005.

[13] G. Gualtieri and M. Tartaglia. Predicting urban traffic air pollution: A GIS framework. *Transportation Research Part D: Transport and Environment*, 3(5):329 – 336, 1998.

[14] K. Hennermann. *Kartographie und GIS. Eine Einführung*. Wissenschaftliche Buchgesellschaft, 2006. ISBN 978-3-534-19692-0.

[15] W. Knospe, L. Santen, A. Schadschneider, and M. Schreckenberg. A realistic two-lane traffic model for highway traffic. *Journal of Physics A: Mathematical and General*, 35(15):3369–3388, April 2002.

[16] S. Krauss, P. Wagner, and C. Gawron. Continuous limit of the nagel-schreckenberg model. *Phys. Rev. E*, 54(4):3707–3712, Oct 1996.

[17] T. Nagatani. Effect of car acceleration on traffic flow in 1d stochastic ca model. *Physica A: Statistical and Theoretical Physics*, 223(1-2):137 – 148, 1996.

[18] K. Nagel. Multi-modal traffic in transims. In *in: Schreckenberg, M.; Sharma, S.D. (eds.), Pedestrian and Evacuation Dynamics*, pages 161–172. Springer, 2001.

[19] K. Nagel, K. Nagel, R. J. Beckman, and C. L. Barrett. Transims for urban planning, 1999.

[20] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2(12):2221–2229, December 1992.

[21] K. Nagel, D. E. Wolf, P. Wagner, and P. Simon. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1425–1437, Aug 1998.

[22] B. Raney and K. Nagel. Iterative route planning for modular transportation simulation. In *in Proceedings of the Swiss Transport Research Conference, Monte Verita*, 2002.

[23] M. Rickert and P. Wagner. Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. *International Journal of Modern Physics C*, 7(308):133–153, 1996.

[24] M. Rickert, P. Wagner, and C. Gawron. Real-time traffic simulation of the german autobahn network. In *Proceedings by World Scientific Publishing Singapore*, 1996.

[25] S. J. Russell and Norvig. *Artificial Intelligence: A Modern Approach (2nd edition)*. Prentice Hall, 2003.

[26] B. Sadoun. On the simulation of traffic signals operation. *Simulation*, 84(6):285–295, 2008.

[27] M. Schüle, T. Bieser, P. Karänke, and S. Kirn. Integration einer Multiagentensimulation in ein Geoinformationssystem. In M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, B. Speitkamp, and P. Wolf, editors, *Multikonferenz Wirtschaftsinformatik*. GITO-Verlag, Berlin, 2008.

[28] M. Schüle, R. Herrler, and F. Klügl. Coupling GIS and Multi-agent Simulation - Towards Infrastructure for Realistic Simulation. In G. Lindemann, J. Denzinger, I. J. Timm, and R. Unland, editors, *MATES*, volume 3187 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2004.

[29] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes. The fundamental diagram of pedestrian movement revisited. *Journal of Statistical Mechanics: Theory and Experiment*, 10:2–+, Oct. 2005.

[30] M. Treiber and A. Kesting. *Verkehrsdynamik und -simulation*. Springer-Verlag Berlin Heidelberg, 2010.

[31] P. Wagner. Traffic simulations using cellular automata: Comparison with reality. In *Traffic and Granular Flow*. World Scientific, 1995.

[32] J. Wahle, R. Chrobok, A. Pottmeier, and M. Schreckenberg. A microscopic simulator for freeway traffic. *Networks and Spatial Economics*, 2:371–386, 2002. 10.1023/A:1020899528337.

[33] F. Weifeng, Y. Lizhong, and F. Weicheng. Simulation of bi-direction pedestrian movement using a cellular automata model. *Physica A: Statistical Mechanics and its Applications*, 321(3-4):633 – 640, 2003.