CELLULAR AUTOMATON MODELING OF THE AUTOBAHN TRAFFIC IN NORTH RHINE-WESTPHALIA

Sigurður F. Hafstein, Roland Chrobok, Andreas Pottmeier, Joachim Wahle, and Michael Schreckenberg, Gerhard-Mercator-University Duisburg, Physics of Transport and Traffic, Germany
Corresponding Author: Sigurður F. Hafstein
Gerhard-Mercator-University Duisburg,
Physics of Transport and Traffic,
Lotharstraße 1, 47048 Duisburg, Germany
Phone: +49 (0) 203 379 3150, Fax: + 49 (0) 203 379 6564
email: hafstein@traffic.uni-duisburg.de

Abstract. In 1992 Nagel and Schreckenberg proposed a stochastic cellular automaton model of vehicular traffic [13], which was able to reproduce some empirically observed non-trivial traffic phenomena like spontaneous traffic jam formation. This publication captured the interest of the physicists community and ever since there has been a continuous progress in the development of cellular automata models of vehicular traffic. The most recent models are able to reproduce free flow, spontaneous jam formation, synchronized traffic, as well as meta-stability. However, these models have one major drawback. They were developed and tested on topologically simple road networks and the translation to large and topologically complex real road networks is non-trivial. In this paper we describe the cellular automaton model we use to simulate the traffic on the autobahn network in North Rhine-Westphalia. Further, we consider some algorithmic implementation details and discuss some of the challenges that arise when using this model on such a huge and topologically complex network.

1. Introduction

Efficient vehicular transport of persons and goods is of vital importance to any modern society. In densely populated areas the capacity of the road network is often to its limits and frequent traffic jams cause a significant economic damage. Moreover, in these areas, it is usually hardly possible or socially untenable to build more roads. An intelligent use of the resource 'traffic infrastructure' is therefore economically crucial. The German state North Rhine-Westphalia (NRW) is an example of such a densely populated area where the capacity of the road network is not able to satisfy the traffic demand during the rush-hours. Every day there are congestions on the autobahns in the Rhine-Ruhr region (Dortmund, Duisburg, Düsseldorf, Essen, etc.) and in the area around Cologne and Leverkusen. To make things even worse, the traffic demand is still growing. For this reason, new information systems and traffic management concepts are clearly needed.

Data regarding the traffic state on the autobahns in NRW are mainly provided by about 4,000 loop detectors and infrared or video detection devices. These devices are locally installed and deliver measured data to central servers minute by minute. The most important measured quantities include *the number of passenger cars passed*, *the number of trucks passed*, *the average speed of the passenger cars*, *the average speed of the trucks*, and *the occupancy*, i.e., the sum of the times a vehicle is on the loop detector. Our approach to generate the traffic state in the whole autobahn network from these locally measured quantities is to feed the data into an advanced cellular automaton traffic simulator. The simulator does not only deliver information about the traffic states in regions not covered by measurement, but also delivers reasonable estimates for other valuable quantities like travel times for routes, a quantity that is not directly accessible from the measurements of the detectors.

In this paper we will discuss the simulation model we use, the data structures for an efficient algorithmic implementation of the model, some challenges encountered when using this model on such a huge and topologically complex network, and the visualization interfaces to the simulated traffic. Firstly, we have a macroscopic visualization interface, implemented as a *Java* applet, which makes it possible to present the simulated traffic state on the autobahn (the freeway network in Germany) in NRW to the public at *http://www.autobahn.nrw.de*. Secondly, we have a microscopic visualization interface, implemented in the *C++* programming language using the *Glut OpenGL* 3D-graphics library, which shows the traffic on an arbitrary piece of the autobahn in 3D.

2. Simulation Model

Because data is fed real-time into the simulator it has to be efficient, that is, at least real time. Due to their design cellular automata models are very efficient in large-scale network simulations [4, 15, 8, 16, 18]. The first cellular automaton model for traffic flow that was able to reproduce some characteristics of real traffic, like jam formation, was suggested by Nagel and Schreckenberg [13] in 1992. Their model has been continuously refined in the last 10 years. The model we implemented in our simulator uses smaller cells in comparison with the original Nagel-Schreckenberg model, a *slow-to-start rule*, *anticipation*, and *brake lights*. With these extensions the cellular automaton traffic model is able to reproduce all empirically observed traffic states. Further, we use two classes of different vehicles, *passenger cars* and *trucks*, where the trucks have a lower maximum velocity and different lane changing rules.
Smaller cells allow a more realistic acceleration and more speed bins. We are currently using a cell size of 1.5 m, which corresponds to speed bins of 5.4 km/h and an acceleration of $1.5\,\mathrm{m/s}^2$ ($0 - 100$ km/h in 19 s), in comparison to $7.5\,m$ in the original Nagel-Schreckenberg model. A vehicle occupies $2 - 5$ consequent cells. By using a slow-to-start rule [1] meta-stable traffic flows are modeled by the simulator, a phenomenon observed in empirical studies of real traffic flows [6, 9, 19]. By including anticipation and brake lights [2, 11] in the modeling, the vehicles not solely determine their velocity in dependency of the distance to the next vehicle in front, but also take regard to the speed and acceleration of the front vehicle .
In the Nagel-Schreckenberg model there is only one global parameter, the probability constant (or dawdling parameter) p, and every vehicle, say vehicle n, is completely determined by two parameters. Its position $x_n(t)$ and its velocity $v_n(t)$ at time t. When the vehicle n decides in the time-step $t \mapsto t+1$ how fast it should drive, it does this by considering the distance $d_{n,m}(t)$, i.e., the number of empty cells, to the next vehicle m in front. The modifications mentioned above of the Nagel-Schreckenberg model imply that we have to add some new parameters to the model. When the simulation algorithm decides weather a vehicle n should brake or not it does not only consider the distance to the next vehicle m in front, but estimates how far the vehicle m will move during this time-step (anticipation). Note, that the moves are done in parallel, so the model remains free of collision. This leads to the effective gap

$$d_{n,m}^{\text{eff}}(t) := d_{n,m}(t) + \max(v_m^{\min}(t) - d_S, 0)$$

seen by vehicle n at time t. In this formula $d_{n,m}(t)$ is the number of free cells between the front of the vehicle n and the back of the vehicle m, d_S is a safety distance, set equal to 6 cells (9 m) in our model, and

$$v_m^{\min}(t) := \min(d_{m,l}(t), v_m(t)) - 1,$$

where $d_{m,l}(t)$ is the number of free cells between the vehicle m and its next vehicle in front l, is a lower bound of how far the vehicle m will move during this time-step.
Brake lights are a further component of the anticipated driving. They allow vehicles to react to disturbances in front earlier by adjusting their speed. Empirical observations suggest [5, 12] that drivers react in a temporal- rather than a spatial-horizon. For this reason the velocity-dependent temporal interaction horizon

$$t_n^S(t) := \min(v_n(t), h)$$

is introduced to the model. The constant h determines the temporal range of interaction with the brake light $b_m(t)$ of the next vehicle m in front. The vehicle n does only react to $b_m(t)$ if the time to reach the back of the vehicle m, assuming constant velocity ($v_n = const.$) and that the vehicle m stands still, is less than $t_n^S(t)$, that is,

$$t_n^h(t) := \frac{d_{n,m}(t)}{v_n(t)} < t_n^S(t).$$

In our model we take h equal to 7 s.
The third modification of the Nagel-Schreckenberg model implemented in the simulator is a velocity dependent randomization, which means that the probability constant p is replaced with a probability function dependent on the velocity of the vehicle. Further, the probability is also a function of the brake light of the next vehicle in front. In every time-step for every vehicle n with vehicle m next in front, the

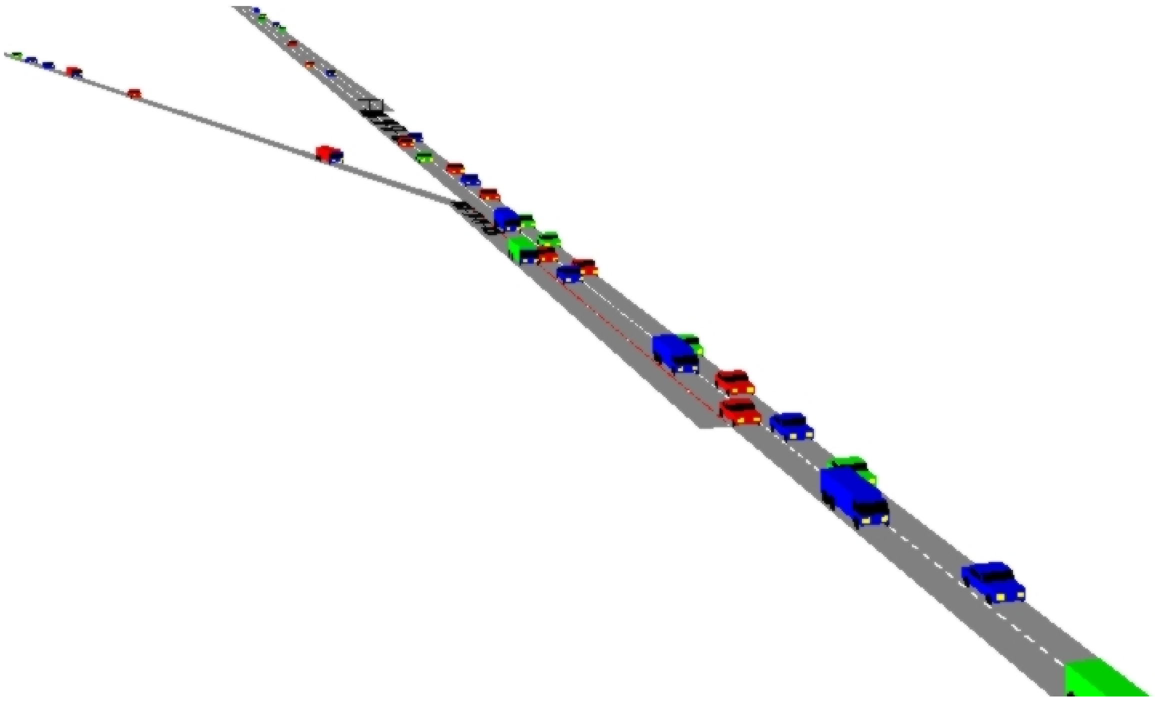
Figure 1: Part of the autobahn in the simulator.

probability that the vehicle n brakes is

$$p = p(v_n(t), b_m(t)) := \begin{cases} p_b, & \text{if } b_m(t) = \text{on and } t_n^h(t) < t_n^S(t), \\ p_0, & \text{if } v_n(t) = 0, \\ p_d, & \text{default.} \end{cases}$$

In our model we take p_b equal to 0.96, p_0 equal to 0.5, and p_d equal to 0.1.
To sum up, to move the vehicles forward in the network the algorithm executes the following steps in parallel for all vehicles n:

- Step 0: Initialization:
  For vehicle n find next vehicle in front m. Set $p := p(v_n(t), b_m(t))$ and $b_n(t+1) :=$ off.

- Step 1: Acceleration:

$$v_n(t+\frac{1}{3}) := \begin{cases} v_n(t), & \text{if } b_n(t) = \text{on or } (b_m(t) = \text{on and } t_n^h(t) < t_n^S(t)), \\ \min(v_n(t)+1, v_{\max}), & \text{default.} \end{cases}$$

- Step 2: Braking:

$$v_n(t+\frac{2}{3}) := \min(v_n(t+\frac{1}{3}), d_{n,m}^{\text{eff}}(t)).$$

  Turn brake light on if appropriate:

$$\text{if } v_n(t+\frac{2}{3}) < v_n(t), \text{ then } b_n(t) := \text{on}.$$

- Step 3: Randomization with probability p:

$$v_n(t+1) := \begin{cases} \max(v_n(t+\frac{2}{3}) - 1, 0), & \text{with probability } p, \\ v_n(t+\frac{2}{3}), & \text{default.} \end{cases}$$

Turn brake light on if appropriate:

$$\text{if } p = p_b \text{ and } v_n(t+1) < v_n(t+\frac{2}{3}), \text{ then } b_n(t+1) := \text{on}.$$

- Step 4: Move (drive):
$$x_n(t+1) := x_n(t) + v_n(t+1).$$

Free lane changes are needed so that vehicles can overtake slower driving passenger cars and trucks. When designing rules for the free lane changes, one should take care of that overtaking vehicles do not disturb the traffic on the lane they use to overtake to much, and one has to take account of German laws, which prohibit overtaking a vehicle to the left. Further, it is advantageous to prohibit trucks to drive on the leftmost lane in the simulation, because a truck overtaking another truck forces all vehicles on the left lane to reduce their velocity and produces a deadlock that may not resolve for a long time [10].
One more variable is needed for the free lane changes, $l_n \in \{\text{left}, \text{right}, \text{straight}\}$ notes if the vehicle n should change the lane during the actual time-step or not. This variable is not needed if the lane changes are executed sequentially, but we prefer a parallel update of the lane changes for all vehicles and that renders this variable necessary. For the left free lane changes the simulator executes the following steps parallel for all vehicles n:

Overtake on the lane to the left:

- Step 0: Initialization:
For vehicle n find next vehicle in front m on the same lane, next vehicle in front s on the lane left to vehicle n, and the next vehicle r behind vehicle s. Set $l_n :=$ straight.
- Step 1: Check lane change:
$$\text{if } d^{\text{eff}}_{n,m}(t) < v_n(t) \text{ and } d^{\text{eff}}_{n,m}(t) < d^{\text{eff}}_{n,s}(t) \text{ and } d^{\text{eff}}_{r,n}(t) > v_r(t), \text{ then set } l_n := \text{left}.$$
- Step 2: Do lane change:
$$\text{if } l_n = \text{left, then change lane for vehicle } n \text{ to the left.}$$

The definition of the gaps $d^{\text{eff}}_{n,s}(t)$ and $d^{\text{eff}}_{r,n}(t)$ is an obvious extensions of the definition above, one simply considers a copy of the vehicle n on its left side. These overtake rules used by the simulator can verbally be summed up as follows: First, a vehicle checks if it is hindered by the predecessor on its own lane. Then it has to take into account the gap to the successor and to the predecessor on the lane to the left. If the gaps allow a safe change the vehicle moves to the left lane. For the right free lane changes the simulator executes the following steps parallel for all vehicles n:

Return to a lane on the right:

- Step 0: Initialization:
For vehicle n find next vehicle in front s on the lane right to vehicle n and next vehicle r behind vehicle s. Set $l_n :=$ straight.
- Step 1: Check lane change:
$$\text{if } d^{\text{eff}}_{n,s}(t) > v_n(t) \text{ and } d^{\text{eff}}_{r,n}(t) > v_r(t), \text{ then set } l_n := \text{right}.$$
- Step 2: Do lane change:
$$\text{if } l_n = \text{right, then change lane for vehicle } n \text{ to the right.}$$

Thus, a vehicle always returns to the right lane if there is no disadvantage in regard to its velocity and it does not hinder any other vehicle by doing so.
It should be noted, that it is not possible to first check for all lane changes to the left and to the right and then perform them all in parallel without doing collision detection and resolution. This would be necessary because there are autobahns with three lanes and more. To overcome this difficulty, the lane changes to the left, i.e., overtake, are given a higher priority than the lane changes to the right. For a systematic approach to multi-lane traffic, i.e., lane-changing rules, see, for example, [14]. For a detailed discussion of the different models see [7, 17, 3] and the references therein.

3. Some Details on an Efficient Implementation

A crucial point in the design of a large scale traffic simulator is the software-engineering part. On the one hand, the chosen data structures have to be abstract enough to model the occurrences in the whole autobahn network and it should be reasonable easy to generalize and make changes in the design. The latter is of great importance, because there is a continuous progress in the theory of traffic flows and the desire for some extensions to the modeling in the future is a certainty. On the other hand, an efficient algorithmic implementation of the dynamics is a necessity for such a large road network. A simulator that is not able to simulate the traffic flow in at least real time during the rush hours is of little value and if it is intended to use the simulator for traffic forecast it has to be multiple real time.
Let us start with the representation of the road network in the simulator. Like in other simulators (e.g., [4, 21]) the network consists of basic elements, links and nodes. A link is a directed bundle of parallel lanes or, more casually, simply a piece of autobahn. A vehicle on a link has local coordinates (cell and lane) with respect to the link. A node is a connection between two links. It stores information about the position of the exit is on the link to be left, about how to leave the link (lane change, drive out of it), and how to calculate the new local coordinates on the target link (cell offset, lane offset). Every link contains pointers to all nodes relevant to it. Further, a link keeps information relevant to the vehicles in every cell. The vehicles have a fast access to this information through their position (cell and lane). A link keeps track of the vehicles on it by a doubly linked list of pointers to the vehicles, see Figure 3. This list is sorted with regard to the cell positions of the vehicles in every time step of the simulation. Note, that this is necessary because the relative order of the vehicles can change due to overtaking. By doing this the vehicles have a fast access to its neighbors, which is essential for the efficiency of the simulator. By combining links and nodes, one is able to build the complex structures of the autobahn network. Examples for these structures are:

- junctions, where vehicles enter or leave the autobahn,
- intersections, at which two autobahns are connected, and,
- triangular intersections, where two autobahns meet, but one ends.

The complexity of an intersection can be derived from Figure 2. Other geometries are rarely found in the autobahn network in NRW. However, they can be constructed easily with the elements used here. Using these links and nodes the autobahn network of NRW was reconstructed. It comprises 3,988 links, 830 on- and off-ramps, and 67 intersections. The overall length of the lanes is approximately 12,200 km, corresponding to more than 8 million cells. The data used for the network were extracted from the *NW*-SIB, a Geographic Information System (GIS) database provided by the state of NRW.
The concept of a vehicle is implemented as a C++ class. The vehicle contains a pointer to the link it is on, its position (cell and lane), the node it is heading to, and various other bookkeeping data. To keep the simulator flexible the most important functions, `CheckLaneChange` and `CalculateVelocity` (see last section), are implemented as static function pointers. An alternative would be to use class inheritance, but this would imply the overhead of virtual functions (every vehicle would carry a pointer to a virtual table where the addresses of the functions to be used is stored), so we decided against it. For efficiency reasons it is crucial to use a custom memory allocation for the vehicles, i.e., redefining operator *new* and *delete* for the vehicles class.
The simulation performs in a multiple real-time on a modern personal computer (Athlon™ XP 1600+, 512 MB DDR-RAM) and in regard to the ever growing computational power it looks promising to combine the simulation with historical data for traffic forecast. We are currently working on this and the preliminary results look promising. A further application for the simulator is to research the influence

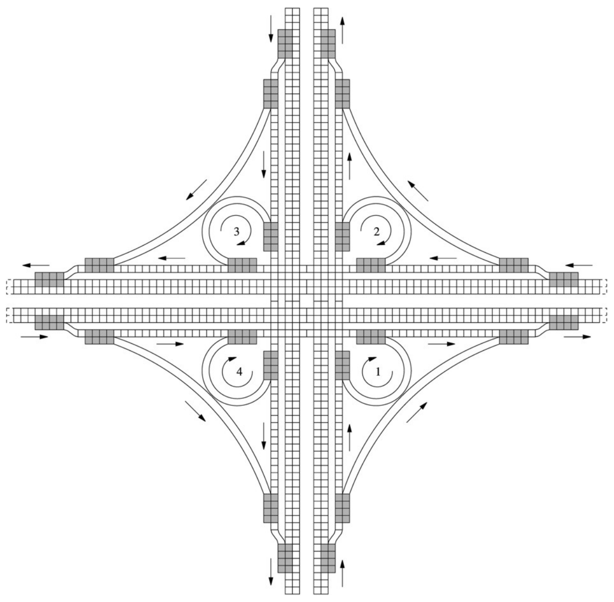

Figure 2: The complex structure of an intersection

of new roads or road-works on the traffic flow.

4. Additional Rules for Complex Real Networks

The cellular automaton model for traffic flow used by the simulator was designed to be able to reproduce the main aspects of the fundamental diagram for real traffic flows (vehicle flow as a function of vehicles per km) and the fundamental microscopic properties, like the time headway distribution. This ability was verified by testing it on topologically simple networks. When simulating the traffic on a large and topologically complex network, like the autobahn network in NRW, some extensions to the cellular automaton model have to be considered. One is the guidance of vehicles and another is a strategy to integrate the measured flow from the loop detectors into the simulation.
A real driver usually has the intention to reach some goal with his driving. This makes it necessary to incorporate routes in the modeling. In principle, there are two different strategies to solve this problem. One can assign an origin and a destination to the road user and then guide him through the network according to this route [15, 16]. For our network origin-destination information with a sufficient temporal and spatial resolution is not available. Therefore, the vehicles are guided in the network according to the probabilities calculated on the basis of the measured data. This means that a vehicle is not guided through the whole network, but every time it reaches a new link it will decide in accordance with the measured probabilities how it leaves the link.
To implement this we use forced lane changes. Forced lane changes are necessary so that the vehicles can drive from on-ramps on the autobahn, from the autobahn on off-ramps, when the autobahn narrows, and when vehicles drive from one particular section of the autobahn on another over an intersection. Forced lane changes differ from free lane changes in a fundamental way. While free lane changes give vehicles the opportunity to overtake vehicles driving slower and thus reduce disturbances, forced lane changes stem from the need to reach a node and are obviously an additional source for disturbances.
The simulator uses gradually increasing harsh measures to force lane changes. At the beginning of an area where a vehicle could change to the target lane, it does so if the gap is sufficiently large and no vehicle is severely hindered. At the end of the area it will bully into any gap regardless of velocity differences. Further, a vehicle driving on its target lane should not leave the lane to overtake. An efficient implementation of this strategy is to store the lane change information in the cells. This gives a fast access through the coordinates of a vehicle. Of course this information depends on the node chosen and whether the vehicle is a truck or a passenger car. Because of this every link has several versions of the lane change information.

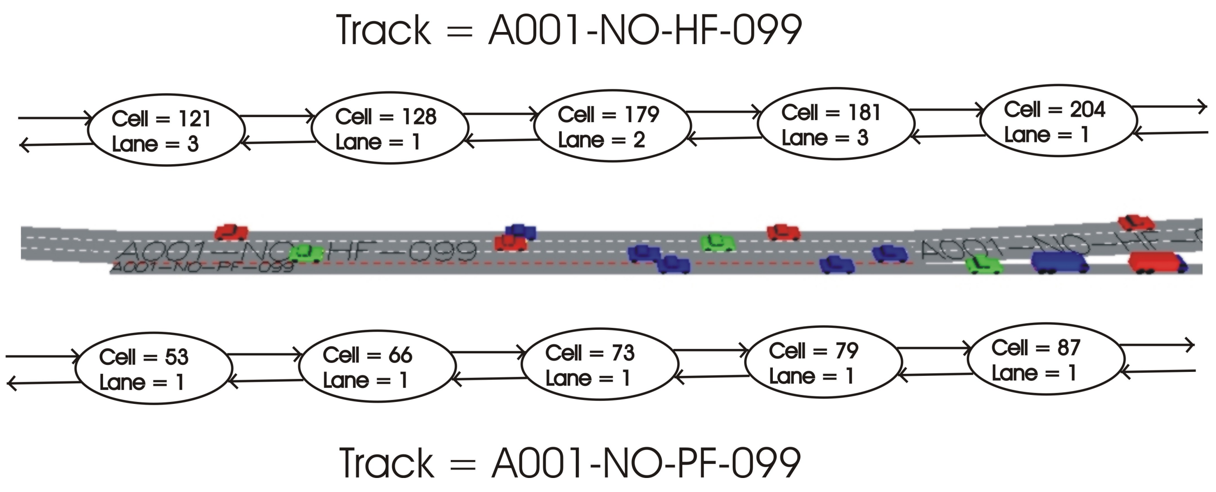

Figure 3: The links (tracks) in the simulator save the vehicles on them as sorted doubly linked lists

To incorporate the real world measurements from the loop detectors into the simulation vehicle-moving, inserting, and removing algorithms have to be applied. This is done at the so-called checkpoints, which are located at those places in the network where a complete cross-section is available, i.e., all lanes are covered by a loop detector. Every time, when checkpoint-data is provided, the simulator uses the measured values to adjust the traffic state in the simulation. The first step is to try to move vehicles behind the checkpoint in front of it and vice versa. If this is not enough to adjust the traffic state, vehicles are inserted or removed. This should be preferred to pure insert/removal strategies, because these can completely fail due to positive feedback if a non-existing traffic jam is produced by the simulation. In this case the simulation measures a low flow in comparison with the real data, so vehicles are added periodically to the ever growing traffic jam leading to a total breakdown.
For realistic results it is further important to minimize the perturbation of the dynamics present in the network due to the data integration. Therefore, we propose a method which follows the idea to add the vehicles to the network “adiabatically”, i.e., without disturbing the system. If the number of vehicles crossing the checkpoint is lower than measured by the detector, vehicles are inserted with regard to the measured mean velocity and mean gap, so that the system is not disturbed, i.e., no vehicle has to brake due to the insertion. This method is therefore called “Tuning of the Mean Gap” [8].

5. The Website http://www.autobahn.nrw.de

The design of the simulator was financially supported by the Ministry of Economics and Small Businesses, Technology and Transport for North Rhine-Westphalia, the reason being, that it wanted a novel web-based traffic information system for the public. This information system is provided by a Java-applet at the URL *http://www.autobahn.nrw.de*. The Java-applet draws a map of NRW, where the autobahns are colored according to the simulated traffic state, from light green for free flow, over dark green and orange for dense and very dense synchronized flow, to red for a traffic jam. Originally, we used yellow instead of orange but color blind persons had difficulties seeing the difference between light green and yellow so we changed to orange. Further, construction areas are drawn at the appropriate positions on the map and their estimated influence on the traffic is shown through red construction signs for a high risk of a traffic jam and green construction signs for a low risk.
The resonance to this novel information system has been very positive and TV-stations, newspapers, and magazines have made positive tests where they compare the actual traffic state to the traffic state presented by our simulation. At the moment we are working on an extended information system, where not only the actual traffic state is visualized, but also a prognosis for the traffic state in 30 minutes. We intend to present this extended traffic information system to the public in Mars 2003.

6. Summary

In this paper we present a microscopic traffic simulator of the autobahn network in North Rhine-Westphalia. The simulator uses an advanced cellular automaton model of traffic flow and adjusts the

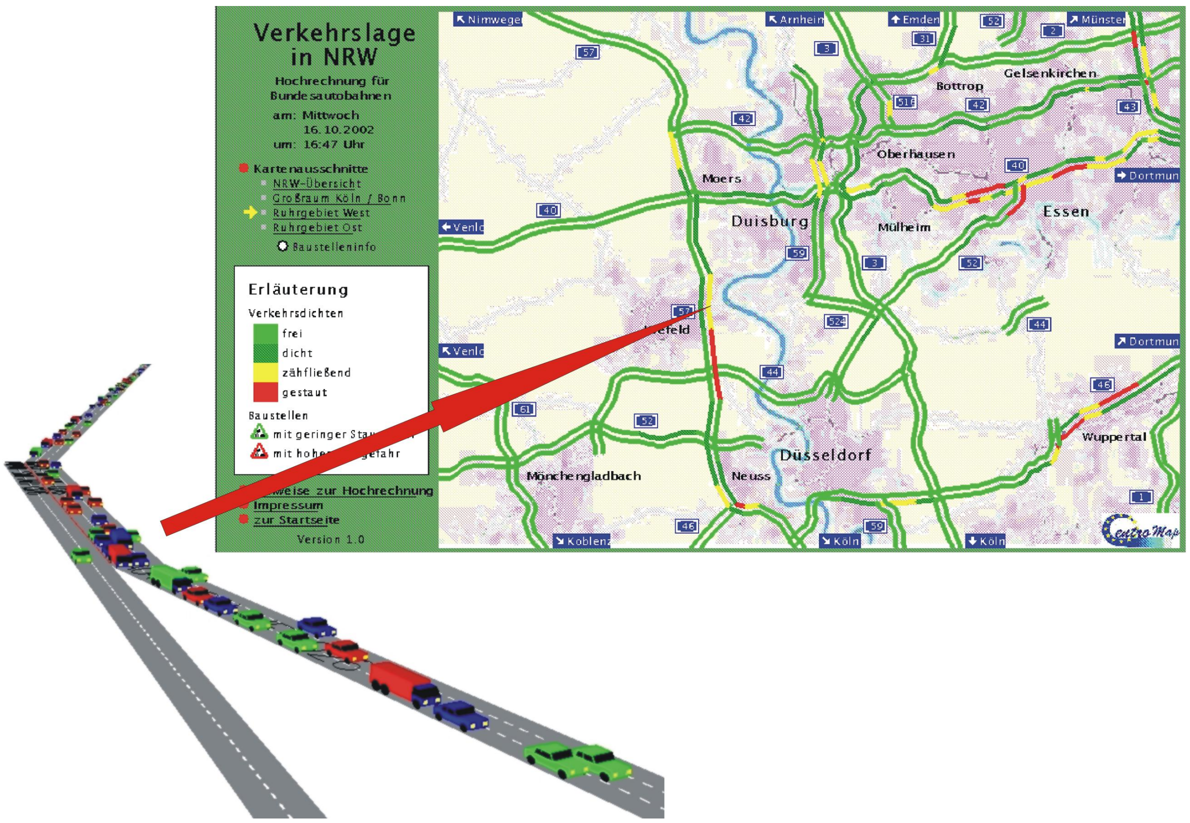

Figure 4: In the Java applet the macroscopic simulated traffic state is visualized

traffic state in accordance with measurements of the real traffic flow provided by about 4,000 loop detectors installed locally on the autobahn. The cellular automaton model, the abstraction of the network, the guidance of the vehicles, and the data integration strategies to periodically adjust the traffic flow in the simulation in accordance with the measured flow on the autobahn were discussed, as well as some details on the efficient implementation of the dynamics and the presentation of the simulated traffic state to the public.
Finally, we are extending the simulator so it can be used for the research of traffic flow control. This is not as simple as it might seem, because any information about the current traffic state available to the public is likely to influence the strategy of the drivers [20].

References

[1] Barlovic R., Santen L., Schadschneider A.,Schreckenberg M. (1998) Metastable states in cellular automata for traffic flow. Eur. Phys. J. B **5**, 793–800

[2] Barrett C., Wolinsky M., Olesen M. (2000) Emergent local control properties in particle hopping traffic simulations. In: Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) (2000) Traffic and Granular Flow '99. Springer, Heidelberg

[3] Chowdhury D., Santen L., Schadschneider A. (2000) Statistical Physics of Vehicular Traffic and Some Related Systems. Physics Reports **329**, 199–329

[4] Esser J., Schreckenberg M. (1997) Microscopic simulation of urban traffic based on cellular automata. Int. J. of Mod. Phys. C **8**, 1025–1036

[5] George H. (1961) Measurement and Evaluation of Traffic Congestion. Bureau of Highway Traffic, Yale University, 43–68

[6] Helbing D. (1996) Empirical traffic data and their implications for traffic modelling. Phys. Rev. E **55**, R25

[7] Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) (2000) Traffic and Granular Flow '99. Springer, Heidelberg

[8] Kaumann O., Froese K., Chrobok R., Wahle J., Neubert L., Schreckenberg M. (2000) On-line simulation of the freeway network of North Rhine-Westphalia. In: Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) (2000) Traffic and Granular Flow '99. Springer, Heidelberg, 351–356

[9] Kerner B., Rehborn H. (1997) Experimental properties of phase transitions in traffic flow. Phys. Rev. Lett. **79**, 4030-4033

[10] Knospe W., Santen L., Schadschneider A., Schreckenberg M. (1999) Disorder effects in cellular automata for two-lane traffic. Physica A **265**, 614-633

[11] Knospe W., Santen L.,Schadschneider A., Schreckenberg M. (2000) Towards a realistic microscopic description of highway traffic. J. Phys. A **33**, L1–L6

[12] Miller A. (1961) A queuing model for road traffic flow. J. of the Royal Stat. Soc. **B1**, 23, University Tech. Rep. PB 246, Columbus, USA, 69–75

[13] Nagel K., Schreckenberg M. (1992) A cellular automaton model for freeway traffic. J. Physique I **2**, 2221–2229

[14] Nagel K., Wolf D. E., Wagner P., Simon P. (1998) Two-lane traffic rules for cellular automata: A systematic approach. Phys. Rev. E **58**, 1425–1437

[15] Nagel K., Esser J., Rickert M. (2000) Large-scale traffic simulations for transport planning. In: Stauffer D. (Ed.), Ann. Rev. of Comp. Phys. **VII**, 151–202, World Scientific, Singapore

[16] Rickert M., Wagner P. (1996) Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. Int. J. of Mod. Phys. C **7**, 133–153

[17] Schreckenberg M., Wolf D. (Eds.) (1998) Traffic and Granular Flow '97. Springer, Singapore

[18] Schreckenberg M., Neubert L., Wahle J. (2001) Simulation of traffic in large road networks. Future Generation Computer Systems, **17**, 649–657

[19] Treiterer J. (1975) Investigation of traffic dynamics by areal photogrammatic techniques. Tech. report, Ohio State University Tech. Rep. PB 246, Columbus, USA

[20] Wahle J., Bazzan A., Klügl F., Schreckenberg M. (2000) Anticipatory Traffic Forecast Using Multi-Agent Techniques. In: Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) Traffic and Granular Flow '99. Springer, 87–92

[21] Yang Q., Koutsopoulos H. N. (1996) A microscopic traffic simulator for evaluation of dynamic traffic management systems. Transp. Res. C **4**, 113–129