Chapter 10

A path-based simulation of the freeway network of Northrhine-Westfalia

This chapter describes simulation experiments using the freeway network of the German Land Northrhine-Westfalia (NRW). Except for the different underlying network, the structure of the simulations is very similar to the last chapter.

This time, there are many travelers with different origin-destination pairs. Travelers have route plans (paths) so that they know on which intersections they have to make turns in order to reach their destinations. Each traveler has a choice between 10 different paths. Each traveler chooses a path, the microsimulation is executed according to the plans of each traveler (no replanning during the trip), and each traveler remembers the performance of his/her option.

Each traveler tries each option once. Afterwards, she usually chooses the option which performed best in the past, except that, with a small probability p_{other}, another option is chosen randomly, in order to update the information about other options.

This approach—giving each agent a set of options and let each agent act on the basis of the performances of these options—is a simplified version of Holland's classifier systems [67]. See [4] for an application of these ideas in an economic context.

In contrast to the last chapter, no congestion pricing will be examined.

10.1 Network

The following simulations are based on a digital version of the freeway network of NRW, where some lower level highways (Bundesstraßen) are added in order to prevent free ends inside the network. The code is written for parallel computers using message passing, in principle for an arbitrary numbers of computational nodes (CPNs). In practice, two Sparc10 workstations, coupled via optical link and using PVM 3.2, were used. That gives the idea that experiments such as the following are already possible with a still modest amount of hardware, and that the consistent use of parallel supercomputers will allow systematic analysis of much larger systems.

The network data which is used comes from Rickert [141] as an intermediate step of his input data preparation. The original data is a list of nodes and a list of edges, where the list of nodes contains all ramps, junctions, and intersections, and edges are the connecting segments. In a first step, Rickert has connected edges to *superedges*, where all nodes of degree two (e.g. ramps) are omitted. In a next step, these superedges are distributed across CPNs, according to a heuristic which yields plausible but not optimal load-balancing and boundary minimization:

- Sort all supernodes according to their y coordinate.
- Cut the network at a certain coordinate y_0 such that all nodes with $y < y_0$ are in one part, and all other nodes are in the other part. Choose y_0 such that the computational load is as balanced as possible. Segments which are cut by this procedure are cut in the middle, and each CPN is responsible for the half of the segment which is adjacent to the local node. By this, one can calculate the length of the network in each new part, and y_0 is chosen such that both new parts have as equal length as possible under the constraints. This is necessary because most of the computational time is needed to perform the dynamical update on the road segments.
- Repeat the procedure with the largest network part which is currently there, using the x coordinates when y ones were used last time (and vice versa). Stop when enough distributed network parts are available.

More details can be found in [141]; since the following simulations were done on two coupled workstations, effectively the result after the first step was used.

Two files are constructed from the result of the network distribution algorithm:

- A list of *supernodes*, which are the nodes which "survived" the above pruning process, i.e. which are junctions or intersections, together with geographical coordinates, and the number of the CPN where they belong to.
- A list of the *superedges*, each superedge together with the numbers of the start and end node, the length of the superedge in traffic model units, and the CPN numbers of both nodes at the ends.

This data is the network basis of the following simulations.

Apart from the network and the individual trip plans, the simulation is kept as simple and straightforward as possible. This includes the following:

- Single directional lanes, i.e. one lane in each direction. The multiplication of the computational load by using multiple directional lanes seems—apart from showing the principal feasibility, which has been done by Rickert—only be justified when one has information or assumptions which need multiple lanes. Examples for this would be the knowledge of the exact number of lanes at all locations, or differences between truck and passenger car trip demand. Since, for this example, no realistic data was available beyond the network data, and since this is the first test of this approach using trip plans inside a realistic network, the use of single lanes seems reasonable. Besides, there are currently, to my knowledge, no results of microscopic simulations of networks of this size with travel plans in the literature at all.
- Oversimplified ramps. The ramp algorithm is the same as in Chapter 9. This means specifically that there are no connecting segments between freeways: A car attempts to change from one freeway segment directly to the other according to the ramp algorithm, and if this is not possible, it stays where it is. It is obvious that this will be overestimating disturbances near intersections. However, in view of the traffic flow results, it is expected that the difference to more realistic intersections is small: Usually, traffic is either sub-critical, where disturbances decay quickly, or supercritical, where disturbances grow.

It might only make a difference exactly at maximum flow. However, it was argued in Chapters 6 and 7 that fluctuations at maximum speed result in occasional disturbances, which leads to an upper cut-off for the critical behavior. At least as a first approximation, it should not make a difference if these disturbances are provided by the driving dynamics or by the intersection dynamics.

10.2 Specific simulation setups

The specific simulation set-up is described in the following. Since no origin-destination data was available, the simulation is a plausible approximation of long distance traffic through NRW. Let us denote by *boundary segments* all segments which lead through the border of NRW and which are thus connected to the rest of the network only at one end. Let us also denote by *boundary nodes* the nodes which connect the boundary segments to the rest of the network.

For the simulation set-up, the boundary segments are replaced by *inflow segments* of equal length $l_{inflow} = 2000$. Then, each site of each inflow segment is occupied by a car. This situation looks like a mega-jam on each inflow segment, similar to Fig. 6.4, but, as we know, it has the same effect as letting enter cars through the boundaries at the maximum flow rate. Each car randomly chooses a destination, which is one of the other boundary segments. The probability to chose a certain destination depends on the air line distance between the origin and the destination segment; three distributions were used:

$$P(destination) = const \ , \qquad (\text{"random"})$$

$$P(destination) \propto distance \ , \qquad (\text{"linear"})$$

and

$$P(destination) \propto (distance)^4 \ . \qquad (\text{"fourth order"})$$

Obviously, "fourth order" is heavily biased towards long trips.

Then, each vehicle gets a list of 10 different paths to reach its destination. These lists have been pre-calculated for all used O-D-pairs, and contain the 10 geometrically shortest paths which do not use the same node twice ([113], see Appendix). In cases where this is not possible, the options are filled up by replications of a possible path. Each vehicle individually now decides which path to use. In the first day (= period), each vehicle uses the shortest path; during the subsequent 9 days, each vehicle randomly selects one of the not yet tested options. Afterwards, it selects, as mentioned in the introduction, the option with the best performance, apart from the probability $p_{other} = 5\%$ to re-test one of the other options, which is then chosen randomly.

After these preparations, the microsimulation starts. Vehicles are updated according to the standard dynamical rules, and they change segments when they are at an intersection. Thus, they enter the network via the the boundary node, where they make their first route decision according to their plans. Each vehicle then follows its plan until it reaches its destination segment, and when it reaches the end of this segment, it notes the arrival time t_{arriv}, i.e. the current iteration step of the simulation, which is used as performance criteria for this specific path. When this is the first time this specific path has been used, this is used as the first guess; for repeated trials of the same route, two schemes are used:

- **Myopic:** Each traveler remembers only the last instance, i.e.

$$t_{remembered}(option) = t_{arriv} \ ,$$

where $t_{remembered}(option)$ is the arrival time this car keeps in memory for this option, and where it bases subsequent decisions on.

- **Averaging:** An exponentially decaying memory:

$$t_{remembered}(option) = const \cdot t_{remembered} + (1 - const) \cdot t_{arriv} \ .$$

A value of $const = 0.7$ was used; $const = 0$ reduces to the myopic scheme.

In practice and on the level of the descriptions below, no differences between both learning schemes were observed.

After all vehicles have reached their destinations and recorded the above information, the simulation is restarted, where all vehicles have the same initial position and the same destination as before, but may choose a different path according to the rules described above.

10.3 Simulation runs

Figs. 10.1 shows an example of a simulation after 400 and 1400 iterations. Iterations are roughly equivalent to seconds (cf. Chapter 3). One clearly sees how the initially empty network is filled by vehicles coming in from the boundary nodes. Each pixel in the plot corresponds to a small region of 30 sites (225 m). Green dots denote that there is at least one car in the region, yellow stars mark slightly over-critical regions, where at least one car has velocity zero, and red triangles mark jams: The density here is above 0.4.

Figs. 10.2 demonstrates the result of the learning algorithm. Both the top and the bottom figure uses exactly the same initial configuration of cars with their individual targets. Targets have been selected according to the "fourth order" probability distribution, thus there are practically only long trips in the simulation. Both figures are snapshots of the situation after 6000 iterations (100 minutes). The top figure shows the situation when every driver follows the geometrically shortest path. Meanwhile, in the bottom figure, drivers act according to their previous experiences, i.e. they usually use the path where they were fastest in the past. Note some important differences between the figures:

- Drivers learn to use A43 from Wuppertal through the Ruhrgebiet. A43 is not contained in any shortest long-distance path.
- At the Kölner Ring, problems are after the learning not only on the eastern but also on the western side. The Kreuz Köln-West is jammed.
- A1 between Wuppertal and the Kamener Kreuz is much more crowded after the adaption, with densities above 0.4 at many places.
- At the Kamener Kreuz, there is now not only a jam for people coming from Hannover, but also for people coming from the Ruhrgebiet.

Generally speaking, people "learn" according to the programmed rules to equilibrate the jams.

For comparison, the same situation is shown one "day" later (Fig. 10.3 top). Moreover, the bottom figure shows a situation (after "learning") for a *linear* distance distribution. Clearly, the overall structure of the jams in both cases remains similar. The fact that there are less jams in the interior of the network for the linear distance distribution is to be expected: The share of short trips is higher, and these trips do not go through the interior of the network.

From these two examples and from further results (not shown), one can conclude that in spite of the stochasticity of the model many of the results are robust, as well from day to day as between different distance distributions.

Obviously, it will be necessary to replace the arbitrary O-D-pattern of these simulations by more realistic data. Yet, some of the network bottlenecks seem generic with respect to transit traffic through NRW: The jams between Wuppertal and Kreuz Kamen are well known, and, as one sees, a consequence of the missing extension of the freeway A4 beyond Olpe. This extension has since long been planned; but it leads through environmentally sensitive areas, and it is thus under discussion if it will ever be built. Note that the simulation methodology presented here can be used to evaluate the utility of such an extension, or what is needed to replace it by improvements along existing paths. Or, which traffic streams have to be reduced in order to manage with the currently existing infrastructure, and how this can be achieved.

The problems near Krefeld are due to the same bottleneck in North-East/South-West direction. It is also known that the Kölner Ring presents a bottleneck.

10.4 Network throughput

As a quantitative measure, the accumulated number of cars which have reached their destination is counted as a function of time. Fig. 10.4 shows one result, for the fourth order distance distribution. Interestingly, the network performance *de*creases after drivers have learned. The probable explanation is that the network becomes crowded in a "balanced" way after drivers learn, whereas before, some parts are overcrowded and some are rather empty. It is, for example, reasonable to believe that, in Fig. 10.2 (top), a path Kreuz Meckenheim - Kölner Ring West - Wuppertal - Kamener Kreuz - Hannover has higher throughput than in Fig. 10.2 (bottom). This is exactly the problem of the C-D drivers of chapter 9, which are trapped by other traffic.

10.5 Discussion

The relaxation algorithm leads to a reasonable distribution of the traffic streams, given the initial origin-destination-assumptions. Thus, this method is capable to do the equivalent of the static equilibrium assignment for a dynamic and congested situation.

It cannot be expected that the simple assumptions yield a completely realistic picture of traffic streams; and for an exact comparison with reality no data was readily available. This will (and has to be) a topic of further research. However, the result should be judged against the input. And in this respect it is perhaps astonishing that already such a simple input leads to a reasonable distribution of traffic streams. Moreover, the traffic patterns after adaption are robust against different statistical distributions for the origin-destination pairs, which supports the expectation that already relatively few information on realistic trip generation will yield rather realistic results.

The simulation experiments show that it is feasible to do large scale traffic simulation from lower level microscopic principles. If one sees the two extremes of modelling as, on one hand, interpolating between known or measured data points from reality, and, on the other hand, as getting data from "first principles", i.e. from structural information, then the approach here views itself more in the second category. By investing enough structural information, such as demographic data and larger network sizes, it should be possible to reach a level of fidelity which is equivalent to the level of fidelity of measurement sensors. This result is well known from simulations of physical, such as fluid-dynamical, systems. However, there is no reason why it should not also hold for socio-economic systems. A result might be that, in the long run, reality measurements should be run through a simulation model for a "consistency check" before they are "believed". If this sounds unusual, one only has to look into the area of weather prediction, where this method has been a standard for many years now [37].

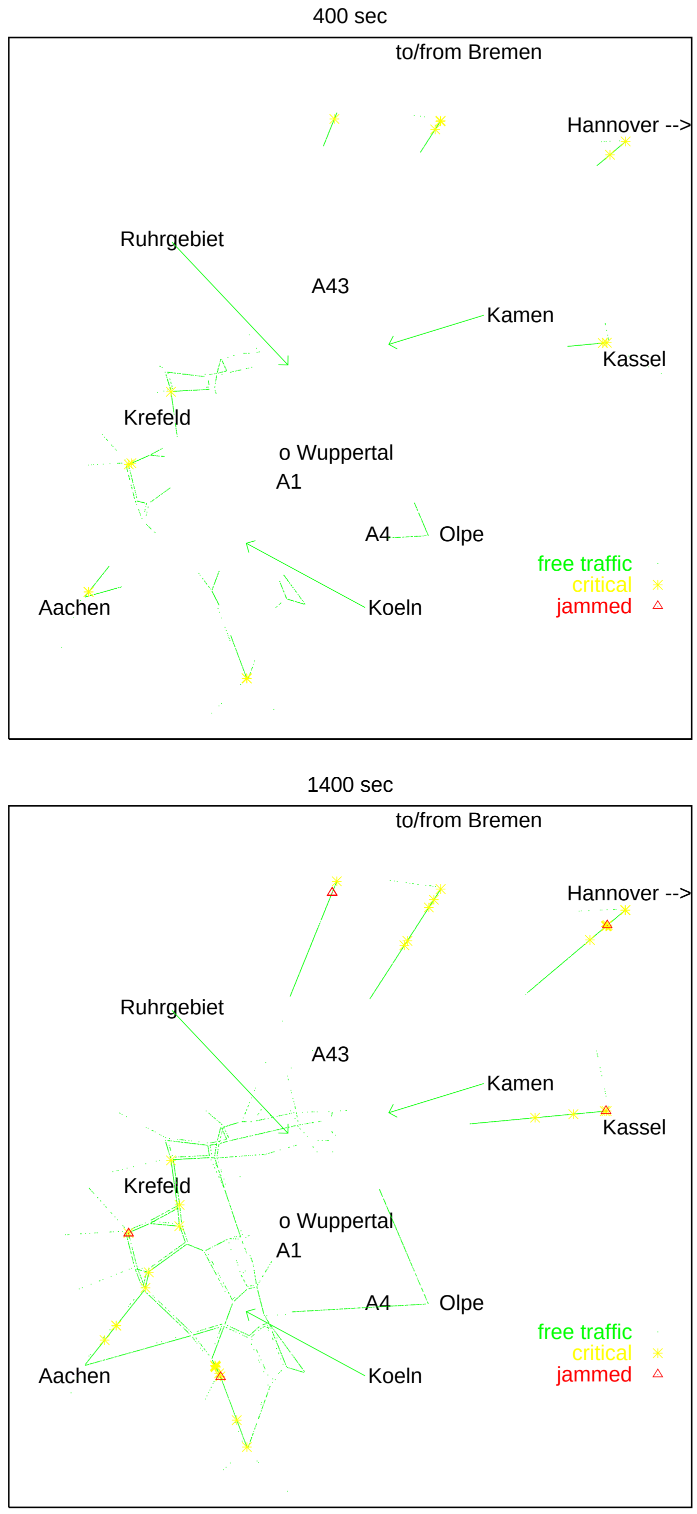

Figure 10.1: Situation after 400 iterations (top) and after 1400 iterations (bottom).

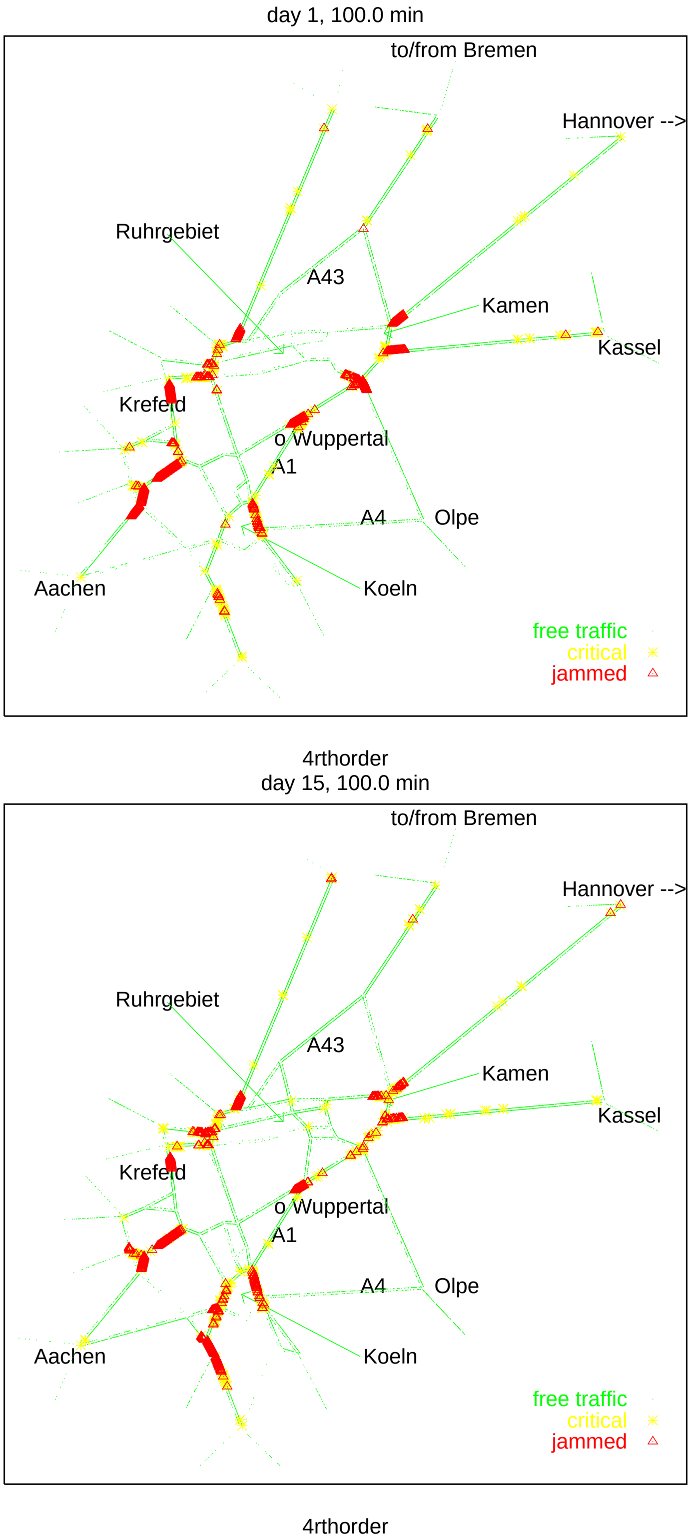

Figure 10.2: *Top:* Situation at the "first day" after 6000 iterations (100 minutes), when trips through the network are chosen with a fourth order preference for long trips, and when all drivers follow the geometrically shortest path. — *Bottom:* Situation at "day 15" after 6000 iterations (100 minutes), for the same initial conditions as for the top figure, but where drivers have "learned".

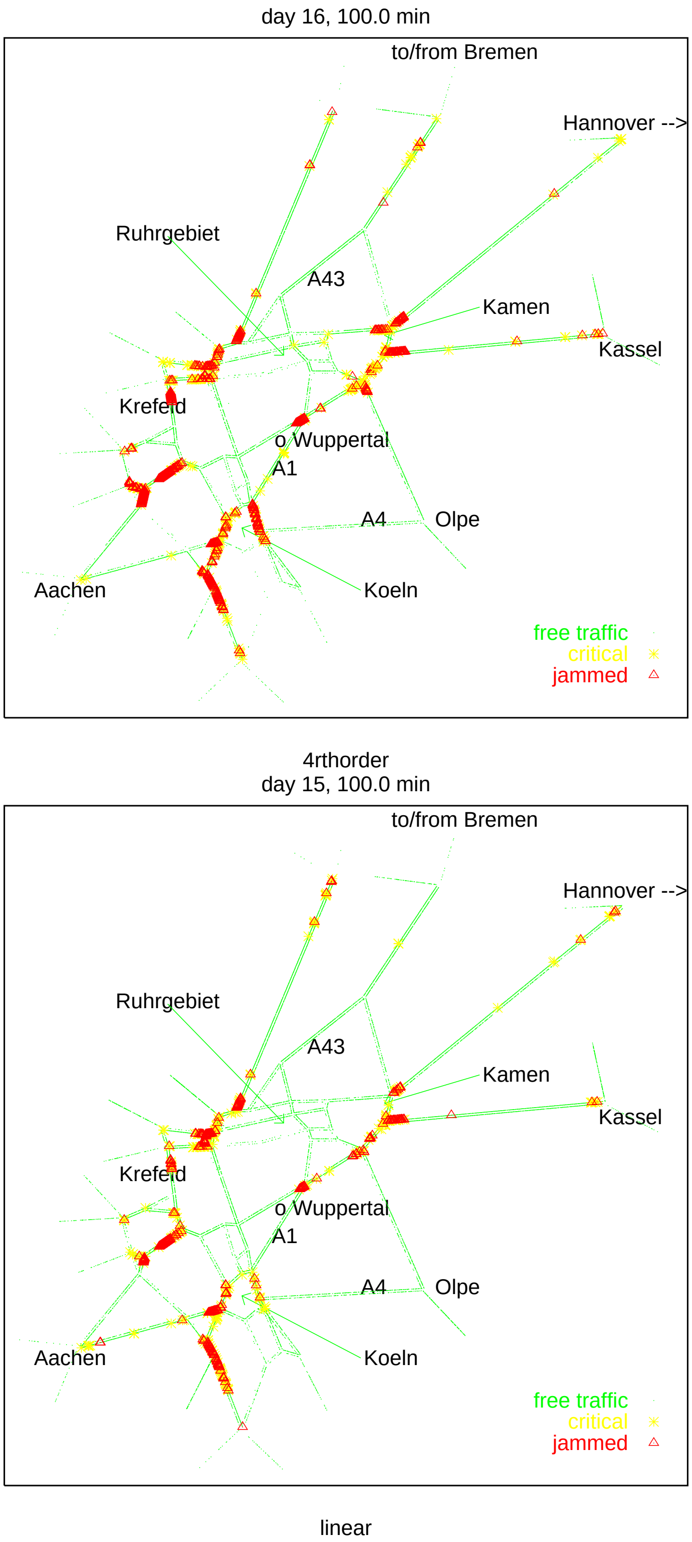

Figure 10.3: *Top:* Same as Fig. 10.2 (bottom), except that it is one "day" later. — *Bottom:* Same as Fig. 10.2 (bottom), except that the distance distribution is linear.

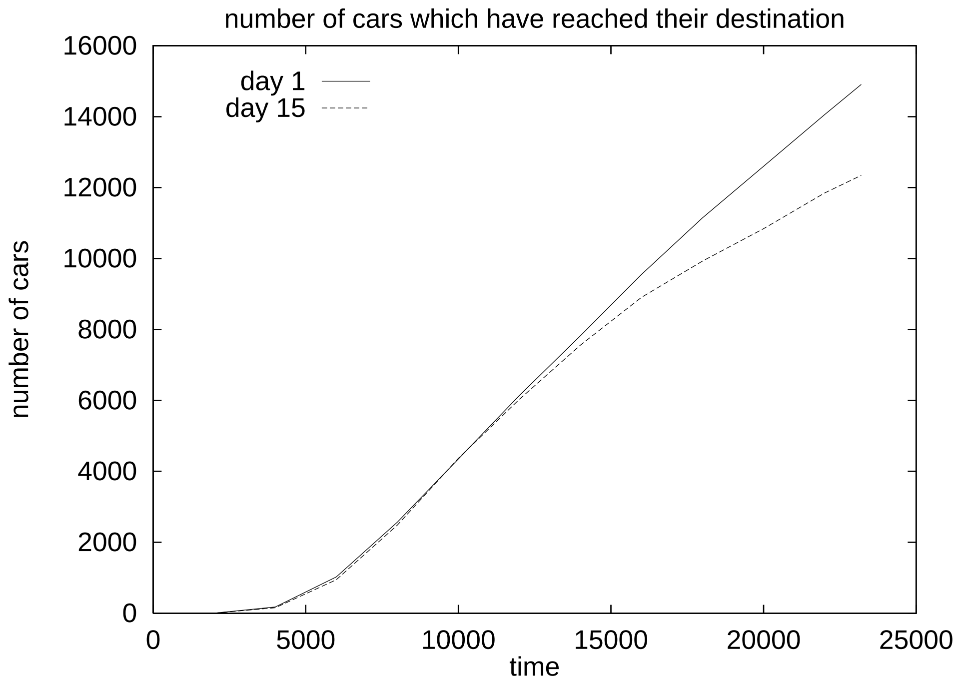

Figure 10.4: Accumulated number of cars which have reached their destination. "Days" 1 and 15 are shown. In day 1, all vehicles drive according to their geometrically shortest path, whereas in day 15 everybody has some knowledge of the travel time on different paths and usually chooses the best one. — Interestingly, the network throughput *de*creases during the relaxation, indicating that indeed something like grid-lock occurs not only in urban traffic, but can also occur in a freeway network.

Part C

COMPUTATIONAL CONSIDERATIONS, OUTLOOK, AND CONCLUSION

Chapter 11

Computational considerations

Much of the research presented in this text relies on computer simulations. Moreover, the ultimate goal is an ultrafast microscopic simulation model for large traffic networks. In such a situation, it makes sense to spend some time thinking about implementation issues. The model is developed in a way that it contains a minimal version of car following logic, and that it would run very fast on vectorizing or parallel supercomputers. To test this, we implemented the model using different algorithms on several different supercomputers.

A question is if one should do such an evaluation at such an early state of the research. Should one not first develop a realistic network model and evaluate the performance of *that* model? The answer for our project is that the early performance evaluation was helpful. The code was still short (20 lines, vs. at least 1200 for a network) which largely simplified the task of adapting to the different computers, the test provided guidelines for which techniques to use in the long run of the project, and it helped keeping the focus on computational speed because it provided benchmarks against which all more extensive models had to compare. For example, a two-lane network implementation of the German freeway network had a network overhead of less than 50%, thus reducing the computational speed by less than a factor of two. Last but not least, it allowed the evaluation against other models with regard of the question if it would make sense to continue this line of research.

This chapter first discusses details of different implementation schemes for the CA model for traffic. Section 2 contains the most important results of implementation tests on different supercomputers. Sections 3 and 4 give more detailed information for different system sizes, followed by a section about issues which came up during a parallelization attempt in the American TRANSIMS project. The last section of this chapter discusses consequences.

11.1 Implementations of the CA traffic model

For the practical coding, we considered three different approaches: site oriented, particle oriented, and an intermediate scheme. *Site oriented* directly implements the CA: A street is represented by a chain $\vec{v}$ of integers with values between -1 and v_{max}. -1 means that there is no particle at this site, whereas the other values denote a particle and its velocity. In contrast, *particle oriented* means that two lists $(x_i)_{i=1,\dots,N}$ and $(v_i)_{i=1,\dots,N}$ contain position x_i and velocity v_i of each particle i ($i = 1, \dots, N$). This is similar to a molecular dynamics algorithm, except that particles are constrained to integer positions and velocities.

Obviously, the particle-oriented approach will always be faster than the site-oriented one for sufficiently low vehicle densities. The particle-oriented approach is more flexible, and since in single-lane simulations passing of vehicles is not possible, the particle lists are always ordered, making efficient codes for all kinds of computers easy to write. In addition, an extension to continuous position and velocity is straightforward (Appendix D).

On the other hand, for the site-oriented (cellular automaton) approach single-bit coding [160] is possible. This means that the model is formulated in logical variables, which may be stored bitwise into computer words. Logical operations on computer words treat all bits of the word simultaneously, giving a theoretical speedup of b, where b is the number of bits per word (usually 32 or 64). However, the *practical* gain for traffic simulations on a workstation is much lower because the bit-oriented approach cannot take advantage of the fact that only a fraction of all sites is occupied by a particle. Nevertheless, we found that, on a workstation, the single-bit algorithm is faster for densities above 0.05 (for $v_{max} = 5$). In addition, the single-bit code runs very efficiently on a Thinking Machines CM-5 using data parallel CM-Fortran and on a NEC-SX/3 traditional vector computer.

Once passing of vehicles is allowed (multi-lane traffic), for the particle oriented approach efficient memory allocation for parallel and/or vector processors becomes more difficult, and single-bit coding for the site oriented approach becomes tiresome. These observations led to a third, *intermediate* approach. As in the site oriented approach, each site is in one of $(v_{max} + 2)$ states, but for the update only the relevant sites are considered. It turns out (see below) that on parallel but not vectorizing computers this algorithm is about as fast as the single-bit version.

In all cases, the parallelization was done geometrically, i.e. dividing the one-dimensional system of size L into p pieces of length $l = L/p$, where p is the number of CPNs (computational nodes). For single-bit coding this is a bit tricky: The standard trick of having the parallel direction(s) different from the bit-coding direction does not work any longer in one dimension. More details can be found in the appendix.

11.2 Computational speeds on different supercomputers

Table 11.1 gives an overview of the computational speeds on selected computers. All values are valid for a vehicle density of $\rho = 0.1$ and system size of $L = 1\,333\,333$ sites, corresponding to 10 000 single lane kilometers. MUPS corresponds to Mega-Updates Per Second, i.e., the number of sites updated per second divided by 10^6. These values are useful in order to compare with other implementations of similar cellular automata or particle hopping systems. The third number is the extrapolated real time system size, i.e. the extrapolated system size (assuming linear speed-up) where the computation would be just as fast as reality. The last number also is an extrapolated real time value, but if gives vehicle seconds per second, i.e. the number of vehicles which could be simulated in real time speed.

The most noteworthy features of the table are emphasized in boldface:

- Although traffic dynamics is quite different, our computational speeds are comparable to those of Ising models: Our fastest value of 533 MUPS on the NEC-SX3/11 may be compared with the value of 1050 MUPS of a very fast implementation of the (two-dimensional) Ising model on the same computer [71].
- On vectorizing computers such as the SX-3 and the CM-5, the single-bit algorithm has a notable advantage over the other algorithms. On all other machines, the intermediate algorithm is never more than a factor 2.5 slower than the single-bit algorithm.

	s.bit (F77)	particle (F77)	intermed. (C)	netw. (C)
Sparc10	0.33 sec 4.0 MUPS 30 000 km 0.4 e 6 veh	 3.4 MUPS	0.71 sec 1.9 MUPS 14 000 km 0.19 e 6 veh	 1.2 MUPS 8 800 km
PVM (5× Sp10)	0.07 sec 19.0 MUPS 140 000 km 1.9 e 6 veh		0.15 sec 8.9 MUPS 65 000 km 0.87 e 6 veh	
SX-3/11$^{(1)}$ 1 CPN	0.0025 sec **533 MUPS** 4 000 000 km 53 e 6 veh		0.48 sec **2.8 MUPS** 21 000 km 0.28 e 6 veh	
GCel-3 1024 CPNs	0.013 sec 102 MUPS 750 000 km 10 e 6 veh	 211 MUPS 1 550 000 km	0.011 sec 121 MUPS 900 000 km 12 e 6 veh	
iPSC 32 CPNs	0.016 sec 83 MUPS 630 000 km 8 e 6 veh		0.038 sec 35 MUPS 260 000 km 3.5 e 6 veh	
Paragon 64 CPNs				 **290 000 km**
CM-5$^{(1)}$ 32 CPNs	0.0077 sec$^{(2)}$ **173 MUPS** 1 300 300 km 17 e 6 veh		0.045 sec$^{(3)}$ **30 MUPS** 220 000 km 2.9 e 6 veh	
CM-5$^{(1)}$ 1024 CPNs				 [> **1.7 e 6 km**]

$^{(1)}$ CPN(s) has/have vector units (SIMD instruction set)
$^{(2)}$ using data parallel Fortran (CMF)
$^{(3)}$ using message passing (CMMD)

Table 11.1: Computing speed of different algorithms on different computer architectures. "s(ingle) bit", "particle", "intermed(iate)", and "netw(ork)" mean the corresponding algorithms described in the text. For each machine and algorithm, the first table entry gives the time each computer needed to simulate a system of size 10, 000 km. From this figure, we derive the other three entries. The second entry is the computational speed in Mega Updates Per Second, and the third and fourth are real time limits in km and in vehicle sec / sec. For further details see text. The entries for network traffic are due to M. Rickert (personal communication).

- Already on a relatively modest machine such as an Intel Paragon with 64 nodes, our real time limit including network handling overhead (intersections etc.) is 280 000 single lane kilometers. Since, e.g., the whole freeway network of Germany is about 60 000 single lane kilometers long

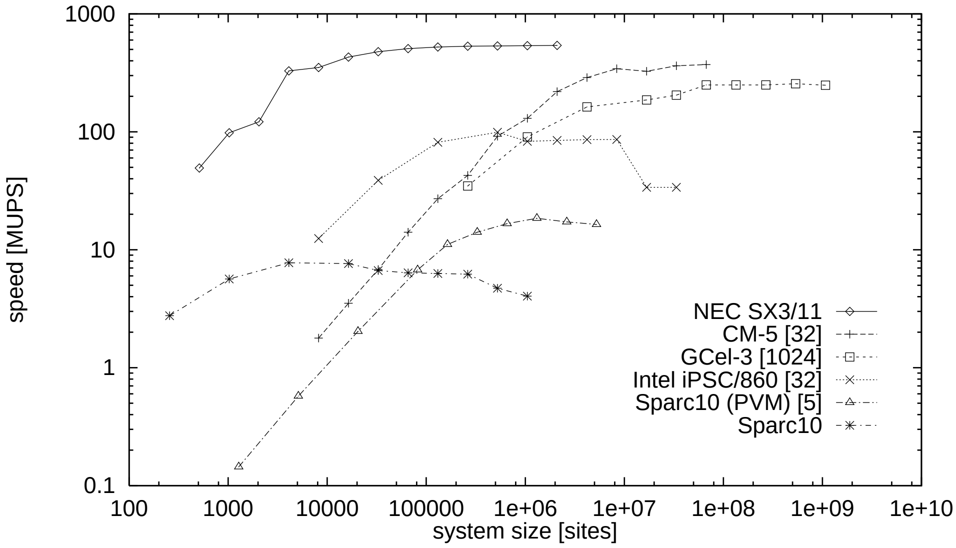

Figure 11.1: Simulation Speeds for different computers as a function of the system size. The computational speed is given in Mega Updates Per Second, i.e. the number of sites which can be updated in one second divided by 10^6. If not noted explicitly, we refer to the computational speed of the whole machine, i.e. all CPNs running simultaneously. Our fastest value of 540 MUPS on the NEC-SX3/11 may be compared with the value of 1050 MUPS of a very fast implementation of the Ising model on the same computer [71].

(12 000 km $\times$ 2 directions $\times$ 2.5 lanes), the real use of this computational speed will be (i) real time applications, where the traffic forecast has to be computed before the situation arrives; and (ii) Monte Carlo simulations, where many runs are necessary.

- The by far fastest "realistic" traffic microsimulation world-wide to date is the PARAMICS microsimulation project [181]. Their real time limit is $\approx$ 20 000 km on 16k CM-200, a machine which is in terms of peak performance slightly faster than a 128-node-Paragon. In other words, it seems that our not completely realistic car following logic buys us about an order of magnitude in performance.

11.3 Different system sizes

The above observations were made for one fixed system size (10 000 km). If one wants to make predictions for different system or computer sizes, one needs more results. This subsection shows some results of systematic measurements.

Fig. 11.1 gives the best computational speed in MUPS (Mega Updates Per Second) we reached for each system size on all the accessed machines.

Fig. 11.2 gives the same information in terms of the "real to simulation time ratio", which is the factor the computer model runs faster than the simulated reality. It is obvious from this figure that for fast simulations of small systems single CPN machines are far better than parallel machines. For example, a system of roads of 100 km could be simulated in 1/40000 of the real time on the NEC vector computer. For slower simulations of large systems, parallel machines become equivalent or even better due to their larger overall memory. Our largest system (on the Parsytec GC-el/3 with 1024 CPNs) had a length of nearly 10 million km single-lane road, which would correspond to 1.6 million km of

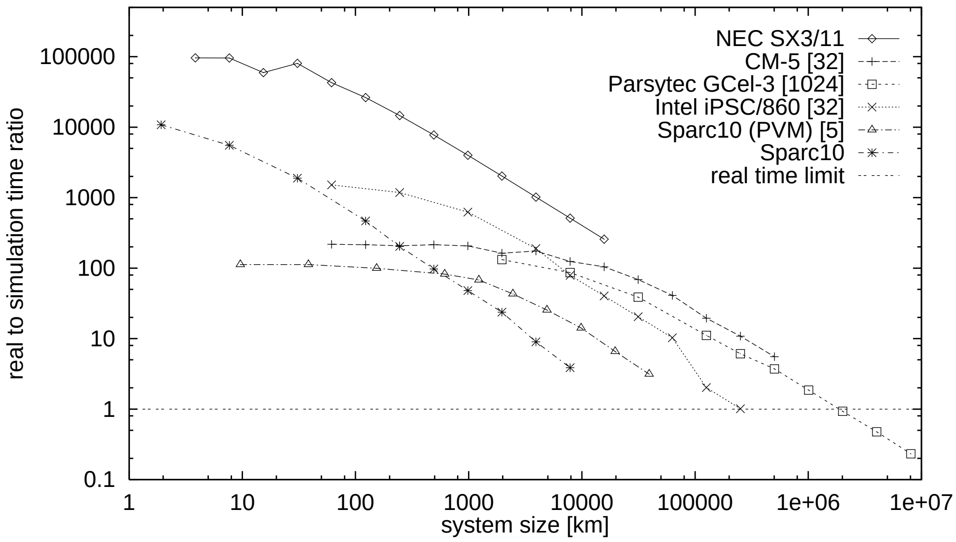

Figure 11.2: Comparison between real time and computer time for different system sizes, i.e. the factor r the computer model runs faster than reality. The system sizes where the curves cross $r = 1$ are the so-called real time limits. Real time limits for current traffic simulation models are near 1000 km ([103, 151]).

freeway with six lanes (three in each direction). In view of practical applications, this result indicates that memory consumption is not a critical issue for microscopic traffic simulations with our approach.

An efficient system size is reached when doubling of the system size demands an approximate doubling of the computer time, which is visible as a slope of -1 in the double-logarithmic plot of Fig. 11.2. For some computer architectures one needs quite large systems in order to make efficient use of the computer. In these cases, adding more processing CPNs to the parallel machine only allows processing of larger systems in the same time (scale-up), but not the same system size in shorter time (i.e., no speed-up). An especially bad case in this respect is the use of coupled workstations ("Sparc10 * 5 [PVM]"), where the single Sparc10-processor outperforms the coupled workstations up to system sizes up to more than 400 km.

The performance gain of the single-bit algorithm against the intermediate one can be seen in Fig. 11.3. The gain is relatively low (between a factor of 1 and 4) on workstation-like CPNs (SUN Sparc 10, Intel iPSC/860) and is completely lost (between 0.4 and 2) on the massively parallel Parsytec GCel–3 with 1024 CPNs. On the other hand, this gain is considerable on the vectorizing machines: We found a factor of about 200 on the traditional vector computer NEC–SX3/11 and still a factor of more than 10 on the CM–5 (the latter only for large system sizes).

11.4 Complexity and scaling

In order to make reliable predictions on scaling behavior, some theory is helpful. Results on a model for the parallel time complexity of the algorithm are found in the Appendix.

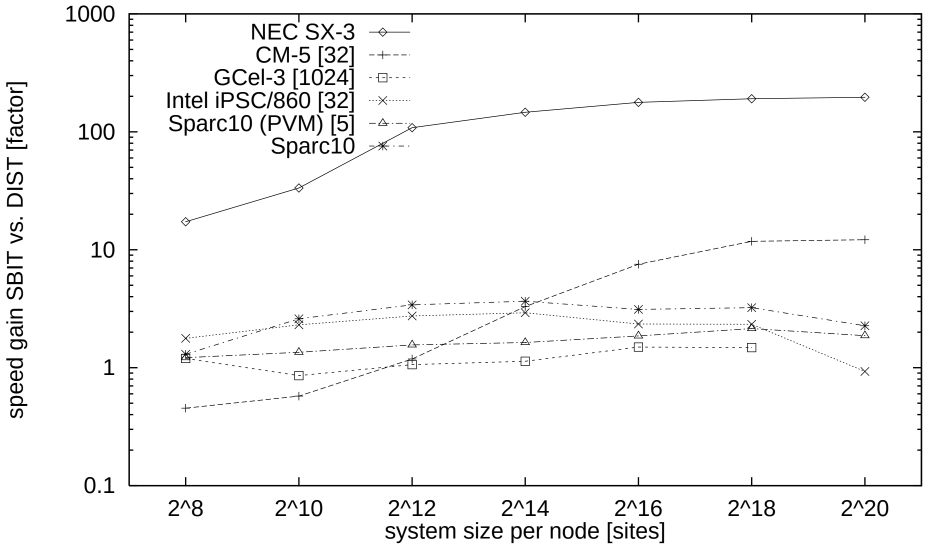

Figure 11.3: Computing speed ratio (SBIT divided by DIST) between the two main coding schemes used for our traffic model as a function of system size. Note the low gain on the massively parallel GCel–3 and the huge gains on the vectorizing computers NEC–SX3/11 and the CM–5 (the latter only for large system sizes).

11.5 A first attempt of parallelizing the TRANSIMS model

During summer 1993, we made an attempt to parallelize an extensive but preliminary microscopic network simulation model of the TRANSIMS project. Without recalling the whole procedure, I want to summarize some of the important results:

- The TRANSIMS demo version was written without planning for a parallel machine. At least in this case it turned out that the structure of the model was not compatible with a parallel implementation. Current status of software tools seems to suggest that *efficient* computer use is only possible when already during the definition of the model and the data structures some hardware realities are taken into account.

- Two technical sub-points of this observation are the following: (i) One should avoid pointers to global objects. The key example here were pointers to network segments, which were undefined as soon as one distributed the network on several CPNs.

  (ii) One should avoid sub-pointers for data which will have to be communicated between CPNs. The problem here was that each vehicle had a pointer to a data field, and some these data fields were again pointers, such as the one to the beginning of the linked list representing the travel plans. Since message passing operations only work on contiguous parts of memory, for the transfer of a vehicle from one CPN to another one had to follow all the pointers and to copy the results into a contiguous memory field, than to send it the the other CPN, and then to inverse the procedure. It looks like a better alternative to keep, e.g., travel plans in arrays of fixed length, even when most of the entries are empty because most travel plans are much shorter.

- As a result of the collaboration, it was decided to try a hybrid simulation model which can switch back and forth between the cellular automaton and a more extensive driving logic. In praxis, this means that even in the CA model, the vehicles have "to carry around" a lot of additional information such as on the travelers in the car, on the car itself, on routing plans, etc. The most efficient implementation of this is that each site in addition to occupation and velocity information also has room for a pointer. When the site is empty, this is a null pointer, when the site is occupied, it is just the starting address of the vehicle data field. And every time when a vehicle moves from a site i to another site j, the address is copied from i to j. It turns out that this slows down the intermediate approach not more than 5%.

- Large programming projects with intelligent objects are only to handle with an object-oriented programming language. Programming large *grid-based* problems (such as fluid-dynamics) in Fortran77 is possible because of the regular program flow, but this is no longer true for other geometries such as road networks. In view of existing languages, C++ is the only portable choice.

  This is even true in view of the fact that the development of C++-compilers and tools lag one to two years behind Fortran on many parallel machines.

11.6 Consequences

Regarding all the information obtained during the implementation experiences, it seems that a bitwise approach will not be useful in the foreseeable future. Vectorizing CPUs are found less and less, and in view of the trend to use standard desktop workstation products together with a more or less powerful communication network to build parallel machines, one would expect that parallel CPNs will only reappear when they become standard in desktop workstations. But without vectorizing CPNs, the bitwise approach is at most four times faster than the intermediate approach.

In addition, the bitwise approach is tiresome for multi-lane traffic, and it does not allow for the additional address-pointer on each site which will be necessary for a hybrid approach. Moreover, for large "intelligent objects" (as opposed to simple grid-based) problems, people will turn to object-oriented programming languages, i.e. C++. One would not expect efficient vectorizing compilers for C++ in the near future even if the hardware existed.

In consequence, most future work in this area in our group will be based on the intermediate approach using C/C++. Throughout this thesis, however, a wider variety of methods was used. Some quantitative data has been obtained using the bit-oriented approach. Most of the simulations in the context of the cruise-control-limit were done using a particle oriented approach, mostly because while not being slower than the bitwise approach the data analysis was easier to code. The network examples at the end use the intermediate approach.

Chapter 12

Outlook

This outlook describes indications for further work, and extensions of the model which have already been investigated by others.

12.1 Validation

12.1.1 2-lane traffic

Rickert [141] has extensively tested a 2-lane model, including algorithms for passing, and including slower vehicles, of maximum velocity $v_{max} = 3$. A typical fundamental diagram is shown in Fig. 12.1. The density of maximum flow has clearly moved to higher densities, as is realistic. Moreover, even for low densities ($\rho < \rho(q_{max})$), the flow q is no longer a simple linear function of ρ. Instead, the following is seen: (i) Very light traffic shows free movement of all vehicles. Average speed $\bar{v}$ here is the average over the maximum speeds of slow and fast vehicles, and flow in this regime therefore $q = \rho \cdot \bar{v}$. (ii) At higher densities, but still below $\rho(q_{max})$, traffic is dominated by the slower vehicles. q therefore grows slower with ρ than in the very light traffic regime.

Moreover, Rickert has shown in simulations that the self-organization of maximum throughput in the outflow situation also holds for 2-lane traffic. Since this is one of the major ingredients for the traffic flow results of Chapters 4 to 7, this supports the conjecture that the traffic flow results are general results independent of the number of lanes.

Latour [92] reports a comprehensive investigation of different algorithms for 2-lane models.

Nagatani [117] also reports two-lane simulations. He uses a model with $v_{max} = 1$ where the velocity update is deterministic.

12.1.2 Acceleration

A remaining problem is the *flow* value at capacity, which is 0.32 vehicles per second. Realistic values are higher (cf. the discussion at the end of Chapter 3). From the traffic flow results, especially in Chapter 6, it is known that the value of q_{max} practically only depends on the acceleration behavior. Therefore, by coding a faster acceleration, it is possible to obtain a higher flow. Seemingly, a problem here is that the acceleration is already rather high: In the average, cars accelerate from 0 to 100 km/h in 10 seconds. However, it turns out that only the acceleration at low velocities really matters.

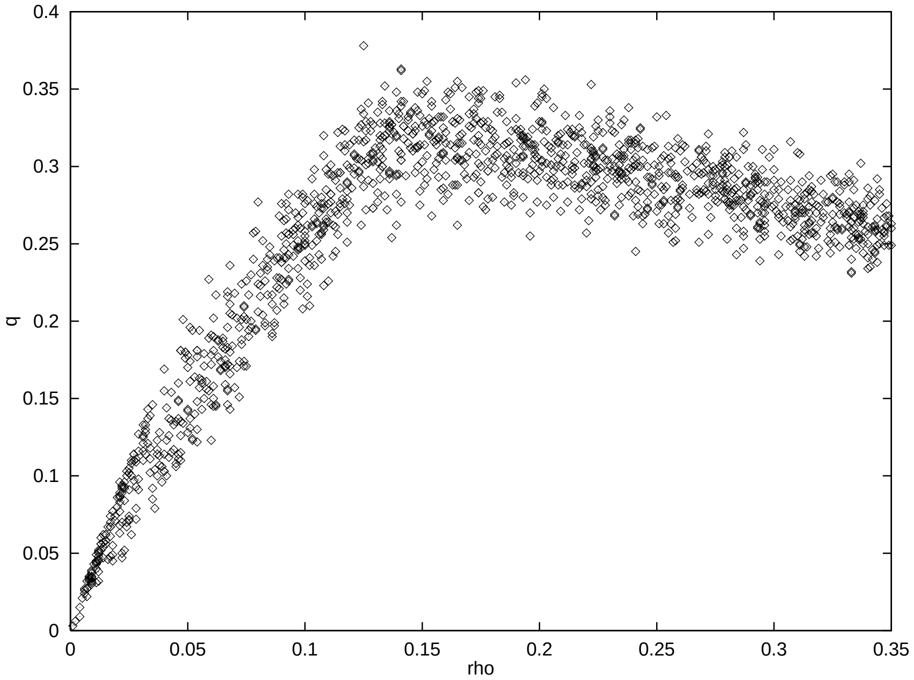

Figure 12.1: Fundamental diagram of simulated two-lane traffic, where 20% of all vehicles are slow (i.e. $v_{max} = 3$). From Ref. [141]

Preliminary results show that, by making the acceleration from $v = 0$ to $v = 2$ faster and making in slower for higher velocities, one can obtain higher flows and more realistic acceleration at the same time. Nonetheless, some more research is needed here. For example, just reducing the probability to accelerate from, say, $v_{max} - 1$ to v_{max}, leads to a wider distribution of possible acceleration times and in consequence to some undesired effects.

Migowsky and coworkers discuss an approach where the car following characteristics of each driver depends on the traffic situation around him/her [111].

12.2 Granular flow

In recent times, granular media have received a lot of attention by physicists [72]. Of main interest for traffic flow is the one-dimensional flow of granular media, e.g. sand. Poeschel [136] reports an experiment and subsequent molecular dynamics simulations of sand falling through a narrow pipe (diameter $\simeq 10 - 50$ particle diameters). Already in a tube of a length of 1.4 meters, clogging occurs and waves develop which are similar to traffic jam waves. Also the mechanism is very similar: Interaction with the rough boundary leads to small fluctuations at maximum speed, which can initiate a chain reaction, leading to a "jam". And the jam is meta-stable when the inflow rate of a jam matches the outflow rate, which is also in the pipe generically true when there are several jams in the system. Lee [96] has done more extensive simulations of the same system and has developed a theory (already mentioned in Chapter 2). Peng and Herrmann [131] use a Lattice Boltzmann simulation method for the system and find in the simulations $1/f^{\alpha}$-scaling of the power spectrum of a particle number time series, which is consistent with experimental measurements [149, 13].

12.3 $1/f$ noise

$1/f$-noise is an ubiquitous phenomenon in nature [36], and in many cases no satisfying explanation is known. Also for traffic flow, $1/f$ noise was observed [115]. The authors analyzed time series of vehicles passing at a given point, i.e. a signal which produced a delta spike every time when a vehicle passed. Pieces of this time-series are combined in a way that the average is removed from the signal, and then the power spectrum (i.e. the square of the Fourier transformation) $S_J(f)$ is calculated. It shows scaling behavior $S_J(f) \sim 1/f$ for low frequencies, and a crossing over to white noise ($S_J(f) \sim const$) for high frequencies.

Such time series analysis should receive further attention in future work: On one hand, time series measurement data is relatively easy to record, and on the other hand, scaling exponents as in the power spectrum are a better starting point for comparisons with CA-like models than most other observables.

The authors' explanation for the $1/f$ measurement is that traffic would follow the noisy Burgers equation, as shown in Chapter 2. The analysis is, however, not consistent with Krug [86], who predicts $S_J(f) \sim f^{-2}$ for the noisy Burgers equation at densities different from $\rho(q_{max})$. Moreover, in view of the results of this work, it seems that the Burgers equation is only valid for rather large length scales. It therefore makes sense to look for other mechanisms which could yield the same result. $1/f$-like behavior was found in two cases, indicating that further work in this area looks promising.

12.3.1 In the cruise control limit

For the CA model, the power spectrum was measured. This was done by first recording the time series for the number of jammed vehicles, $N_l(t)$, in a small segment of length l of a closed system, and then taking the square of the Fourier transform:

$$S_N(f) = |N_l(f)|^2 = |FT[N_l(t)]|^2 \; .$$

In practice, the routine `spctrm.c` of the "Numerical Recipes" [137] was used.

Since the jams have a finite drift velocity, the distribution of hole sizes in space, which has been explained in Chapter 5, is translated into the same distribution of time intervals for the activity. In particular, the hole size distribution in space translates to the first return time for jammed vehicles when sitting at a fixed position in space. It has been shown [107] that given a distribution of first return times of activity $P_{first}(t) \sim t^{-\tau_{FIRST}}$, the power spectrum scales as $S(f) \sim 1/f^{\tau_{FIRST}-1}$. Using the result $\tau_h = \tau_{FIRST} = 2$ this gives precisely a $1/f$ power spectrum for the noise. The power spectrum for the CA model with the parameter $p_{free} = 0.5$, 0.005, 0.00005 and system sizes $L = 10^3$, $L = 10^4$, and $L = 10^5$ was measured in a closed system near the critical density. As shown in Fig. 12.2 for $p_{free} = 0.00005$, $L = 10^5$, and $\rho = 0.072$, the numerical results are in general agreement with this prediction.

12.3.2 In a continuous case

In the appendix, a variant of the CA traffic model using continuous variables is described. This model displays, in a follow-the-leader situation, behavior similarly complex as the CA traffic model. Here, the first vehicle has a lower maximum velocity than all following vehicles. Following vehicles catch

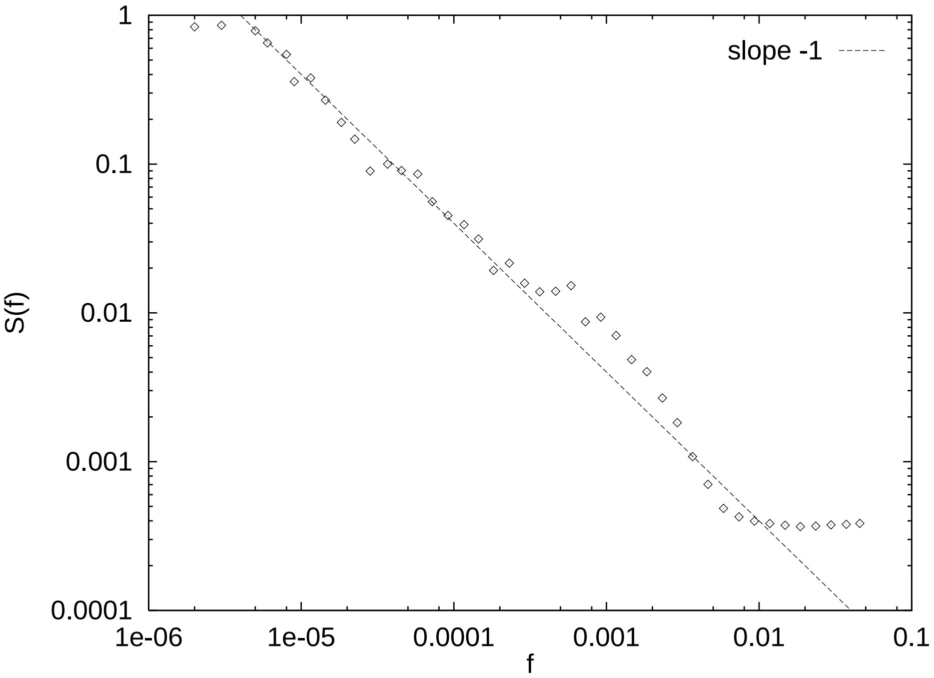

Figure 12.2: Power spectrum, $S_N(f)$, smoothed by averaging, for a closed system of length $L = 10^5$ and with $p_{free} = 0.00005$. Straight line has slope -1.

more and more up. Then, spontaneously and unpredictably, one vehicle slows down and initiates a chain reaction, which looks similar to the traffic jams produced by the CA model.

In a simulation of this model using $N = 3\,000$ vehicles, in every time step the number of cars in a small segment of length l at a position 1000 sites behind the leading car was recorded:

$$N_l(t) = \sum_{i=i_0-1000}^{i_0-1000-l} \delta_i \ ,$$

where i_0 is the position of the leading vehicle, and δ_i is 0 (1) when site i is empty (occupied). Then, the power spectrum of this time series is taken as above.

Also this measurement (Fig. 12.3) shows, similarly to the measurement above, non-trivial behavior, i.e. neither white noise ($S(f) \sim const$) nor $1/f^2$ scaling.

12.4 Phase separation

It is in some sense obvious [114] to try a description of the traffic flow results in terms of a phase separation [99]. Traffic flow at all densities has a tendency to separate into jammed regions and free flow regions, *but only above $\rho(q_{max})$ the jammed regions are stable.* The equilibrium is dynamic, meaning that jams are only maintained because inflow match outflow rates. The transition should then be similar to a gas-liquid transition, except that, at the transition point, the "liquid" (jammed) clusters in the CA traffic model have no preferred size, which makes the transition critical.

This is consistent with the 2-fluid approach [62, 102] which works rather well for traffic flow. In the 2-fluid approach, it is assumed that traffic flow can be approximated by having one portion moving with

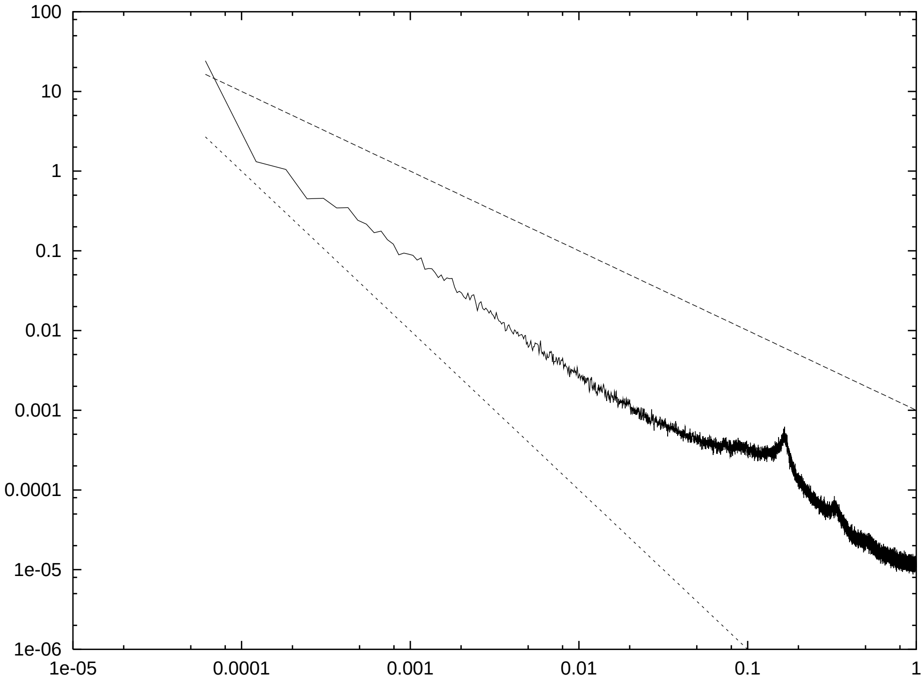

Figure 12.3: Power spectrum, $S_N(f)$, for the continuous model, at a fixed distance of 1000 sites behind the leading vehicle. For comparison, the lines have slope -1 (broken) and -2 (dotted).

maximum velocity and the rest not moving at all, i.e. that all densities above $\rho(q_{max})$ spontaneously separate into portions moving at v_{max} and other portions not moving at all. Note that the 2-fluid approach uses only the average of the phase separation dynamics.

12.5 A dynamic systems approach to transportation simulation and optimization

A general reason for a simulation to be *microscopic* is the following: As soon as simulations are no longer only used as interpolations between known regimes but as a tool to predict new situations (such as the addition or the closing of a road), *structural* (i.e. lower level) information, such as behavioral rules of intelligent objects, is much more robust than aggregated information, such as turn counts. This experience is also known from the simulation of other socio-economic systems, such as of the savings and loans dynamics of building societies [7, 79, 178].

A major reason for doing *high speed* simulations is that it makes a dynamic systems approach easier. One can analyze many different scenarios, or, in more technical terms, systematically investigate the state space of the model (which usually leads to a better understanding of the real system).

Thus, fast microsimulations provide an efficient and robust testbed for the transportation system. This is especially important because experiments with the real system are costly or infeasible. Some examples for testbed applications are:

- Development of efficient optimization techniques for traffic assignment in the non-static case. Fast computational speed means that the model can be used in an iterative loop with methods of (combinatorial or other) optimization, thus providing cost/fitness functions from a relatively realistic simulation instead as from often rather unrealistic mathematical function approximations.

- Testing scenarios such as for adding or closing roads or for the introduction of traffic information or management systems on a large scale basis. This is important for cost-benefit evaluations of such actions, and is equally useful for public service as for private companies which want, for example, invest into individual routing services.
- Evaluation of O-D estimation methods. The simulation can produce dynamic scenarios, one extracts time-dependent turn-count data, the O-D method gives *its* estimate of the O-D matrix, which can then be checked against the simulation input.
- Incident detection algorithms. Incident detection is not possible from single sensors, but only when data input from multiple sensors does not match expectations. For example, a blockage by an accident leads to too few vehicles at sensors downstream of the accident. Such algorithms can be pre-tested in a microsimulation test-bed [151, 165].

In addition, there is another reason why robust microsimulations become important when optimizing the traffic system: Especially near efficient performance, the traffic system (and supposedly many other systems) shows high fluctuations which can be treated using the formalism of critical phenomena (Chapter 9). These fluctuations and the resulting unpredictability provide an effective limitation for optimization attempts. The fluctuations can only be seen when one uses stochastic microscopic simulations, and they can only be represented correctly when the simulation is robust with respect to these fluctuations.

In addition, when fluctuations become important (i.e. when the system becomes more and more efficient), deterministic simulations start to give wrong answers. To recall an earlier example: When the fluctuating back-propagation of a jam triggers a network breakdown with a probability of 5%, then the effect of the network breakdown is only considered in the average when a stochastic model is used.

12.6 A dynamic systems approach to simulation

It is only a small step beyond this to see a simulation itself as a dynamical system [11, 12]. The question is in how far the effects one sees are caused by the dynamics of the simulated system or by the simulation itself.

The traffic dynamics in this text are (hopefully) an image of the traffic dynamics happening in reality. Meanwhile, in future work, a hybrid approach between the CA model and a more extensive microscopic model are planned as part of the TRANSIMS project. This hybrid model is planned to run on parallel machines, and it is planned to independently allocate resources between CA and extensive simulation and between computational nodes of the parallel machine, depending on the question which the simulation is supposed to answer. In this situation, the simulation definitely gets a dynamics of its own, and it is highly probable that even for the same underlying traffic system, two different questions can lead to two different system evolutions because the resources are differently allocated. It will be necessary to identify which of these effects are due to the simulation dynamics and which are due to the traffic system dynamics; and in which cases the answers are reliable and when not.

Chapter 13

Conclusion

Here, I suppose, everybody wants to read the final reason for traffic jams and how to avoid them. This will not be possible; all results indicate that the opposite is true.

This work investigates the usefulness of very fast, hardware-oriented microscopic models for traffic simulation. For this purpose, one of the best available techniques for fast microscopic simulation—the methodology of cellular automata (CA)—is adapted to the needs of traffic simulation. Streets are broken into boxes, and each box can either be empty, or occupied by exactly one car. Movement happens by "jumping" an integer number of boxes, where $v_{max} = 5$ is the maximally possible jump. Rules for acceleration and for braking constitute the dynamics. This is a fairly simple stochastic particle model, which is, especially in view of its simplicity, astonishingly realistic. It correctly reproduces start-stop-waves and fundamental diagrams (relation between average density ρ, flow q, and velocity v). This model is single-lane; Rickert [141] shows that the method is extensible to multi-lane traffic.

Two limits of the model are investigated in detail—the deterministic limit, where all randomness is taken out of the model; and the "cruise control limit", where only the randomness at full speed is set to zero.

The completely deterministic limit is equivalent to a linear, deterministic, and non-diffusive generalized Burgers equation, which is the same as the Lighthill-Whitham theory for kinematic waves with a piecewise linear "inverse V" flow-density relation. Discretizations of the Lighthill-Whitham equations are, together with piecewise linear flow-density relations, indeed in use for traffic simulation.

The cruise control limit allows stable homogeneous flows up to a certain density $\rho(q_{max,det})$. Yet, there is another important density $\rho_c < \rho(q_{max,det})$ with the following property: (i) For $\rho < \rho_c$, a disturbance decays in the average. (ii) For $\rho > \rho_c$, a disturbance (jam) grows in the average. (iii) For $\rho = \rho_c$, in the average, a disturbance does neither grow nor shrink; individual jams fluctuate according to an unbiased random walk. In consequence (Chapter 5), jams at $\rho = \rho_c$ are critical clusters in the sense of critical phenonema. ρ_c is also the density of maximum flow *when starting from random initial conditions*, i.e. $\rho_c = \rho(q_{max,rnd.ini})$. Homogeneous traffic between ρ_c and $\rho(q_{max,det})$ is just *barely* stable: The slightest disturbance leads to a growing jam (according to (ii)); and the outflow from this jam is *exactly at* ρ_c (self-organized criticality).

A third special case is to set the maximum velocity v_{max} to one. This turns out to be comparable (but not equivalent) to the well-known stochastic asymmetric exclusion process (SAEP). In the SAEP, the update is random sequential, which means that randomness enters differently than in the CA traffic model. Nonetheless, the case $v_{max} = 1$ of the CA traffic model behaves similarly to the SAEP, including the effect that kinematic waves move forward for densities below the point of maximum flow $\rho < \rho(q_{max})$, and backward for higher densities $\rho > \rho(q_{max})$. This similarity vanishes with higher v_{max}, where forward moving waves do not exist. The SAEP is, in the hydrodynamic limit, equivalent to a generalized Burgers equation with a linear plus a nonlinear term, which is in turn, after discarding noise and diffusion, equivalent to the Lighthill-Whitham theory using a quadratic flow-density relation (called Greenshields relation in traffic science).

The complete CA model behaves similarly as the cruise control limit, except that *all* configurations above $\rho(q_{max,rnd.ini})$ are only meta-stable, eventually and spontaneously separating into jammed regions and regions of $\rho(q_{max,rnd.ini})$. The latter are critical in the sense described above ("cruise control limit"). However, they are no longer *exactly* critical: There is an upper cut-off in space and time, and the conjecture from all results, especially from the $v_{max} = 1$ case, is that above this cut-off Burgers-like behavior takes over. Judging from the knowledge that single jams definitely exist in reality, and comparing with simulation results, this length scale has to be above 10 km.

This (phase) separation cannot be described by the Lighthill-Whitham theory; fluid-dynamic theories for traffic capture it only after including a momentum equation. These fluid-dynamical theories, contrary to stochastic microscopic models such as the CA, still do not capture the fluctuations around the average behavior.

For all particle models described above it is true that the outflow from a jam automatically selects the state of maximum flow, i.e. $\rho = \rho(q_{max,rnd.ini})$, which means that the critical state is selected generically.

In addition to the effects summarized so far, traffic jams in the model branch. This leads to a fractal appearance; measurements of the distribution of the holes between jams show that the structure is *barely* non-fractal. The structure could explain $1/f^{\alpha}$ noise in measurements, especially since it also shows up under rather different circumstances in a car-following model with continuous variables.

A big advantage of having invested into understanding the exact behavior of the particle model is that one can now argue in how far the model is equivalent to the well-known Lighthill-Whitham model for traffic flow. That is, the argument that the car-following logic used for the model would be too rudimentary to describe realistic traffic, is, besides by the results, also set off by the fact that the macroscopic behavior is similar to, but goes beyond, a well-accepted theory.

Moreover, the critical behavior as indicated above can be described by critical exponents. Since, in addition, there is a phenomenological theory for these exponents, it can be expected that these exponents are universal for microscopic traffic models, i.e. that they do not depend on the exact properties of the models. It can therefore be expected that more complicated models will not show a different behavior in these properties. — Yet, the model presented here is minimal in the sense that further simplifications lead to the loss of essential features such as stochastic jam dynamics.

The second part of this text describes experiments with traffic in networks. In Chapter 9, a simple network with a bottleneck and a longer route around the bottleneck is analyzed. In these simulation experiments, congestion pricing turns out to be useful for a fairer allocation of the resources (road capacity). Although this is generally to be expected, the situation is unclear in the heavily fluctuating environment which traffic near maximum flow is: The toll was only allowed to adapt much slower than the fluctuations, meaning that economic theory would be valid at best in the average; and holes in queues make state estimation problems, e.g. for setting the toll, difficult.

In addition, the simulation serves as an example of a more general problem: The traffic flow results say that maximum flow is close to a critical point, meaning that fluctuations are large. This is confirmed by measuring (in the simulation) also a more practical quantity, the relative prediction error for trip times, which is maximal near maximum flow. Since all traffic management (ATMS-Advanced Traffic Management Systems), such as congestion pricing, will push traffic near maximum flow, predictions and therefore control will become harder and harder. This general conjecture is confirmed by the simulation with the simple example network.

Chapter 10 describes simulations of the freeway network of the German Land Northrhine-Westfalia, with more than 60 000 vehicles simultaneously in the simulation. Building in part on the work of Rickert, travel plans are implemented, meaning that every car knows, when entering the system, which route it wants to take. In the simulations, cars are initially lined up on all outside nodes; they randomly select another outside node as target, and they get the set of the 10 shortest paths to their target to operate on. As with the simple network, the simulation is repeated over and over again, and after having tried out all 10 paths, during the following repetitions each individual tries to find the best path for him/herself.

The traffic then quickly relaxes towards a relatively robust pattern, where bottlenecks, such as the Kölner Ring, are equally congested on all options. Interestingly, the *performance* of the network visibly decreases during this “bottleneck equilibration process”. The result indicates that it is indeed more efficient to “lock up” some portion of the traffic somewhere outside the main network.

Despite the stochastic model, the results are relatively robust as well against day-to-day variations as against changes in the distance distribution. This robustness supports the claim that, to a certain extent, input data from reality measurements such as turn counts can indeed be replaced by structural data, such as plausible assumptions about paths.

Chapter 11 gives an overview of computational performance: Single loop, single lane implementations of the CA traffic model on many different supercomputers give indications for future work and expectations, which are confirmed or exceeded by Rickert’s 2-lane network implementations.

It is hoped that this work contributes to simulation-based dynamics system approaches to socio-economic systems, using microscopic resolution, Monte-Carlo methods, and high performance computing, and combining combinatorial and adaptive methods for modeling human behavior in the system and for eventually optimizing the performance of the system.