Large Scale Traffic Simulations

Kai Nagel[1,2], Marcus Rickert[1,3], and Christopher L. Barrett[1,2],

[1] Los Alamos National Laboratory, TSA-DO/SA MS M997, Los Alamos NM 87545, U.S.A., kai@lanl.gov, barrett@tsasa.lanl.gov, rickert@tsasa.lanl.gov
[2] Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe NM 87501, U.S.A.
[3] Zentrum für Paralleles Rechnen ZPR, Universität zu Köln, 50923 Köln, Germany

Abstract. Large scale microscopic (i.e. vehicle-based) traffic simulations pose high demands on computational speed in at least two application areas: (i) real-time traffic forecasting, and (ii) long-term planning applications (where repeated "looping" between the microsimulation and the simulated planning of individual person's behavior is necessary). As a rough number, a real-time simulation of an area such as Los Angeles (ca. 1 million travellers) will need a computational speed of much higher than 1 million "particle" (= vehicle) updates per second. This paper reviews how this problem is approached in different projects and how these approaches are dependent both on the specific questions and on the prospective user community. The approaches reach from highly parallel and vectorizable, single-bit implementations on parallel supercomputers for Statistical Physics questions, via more realistic implementations on coupled workstations, to more complicated driving dynamics implemented again on parallel supercomputers.

1 Introduction

Nobody likes traffic jams. Yet, they are only the most visible feature among a variety of related problems: Subways which fail to go where or when you need them; pollution of inner cities; etc. Many of these are results of poorly designed transportation systems. However, what is a good design? In our complex world, such a question is not easy to answer. Addition of new streets may *increase* congestion by concentrating formerly spread-out traffic onto one through route [1]; introduction of a transit system may *increase* pollution by making the car which previously was taken to work now available for short trips all with a cold engine [2]; a transportation infrastructure investment payed for by a certain group may actually turn out to benefit a completely different sub-population (winner/looser analysis); a new major arterial meant to relieve congestion may attract new developments along this new arterial, making congestion worse in the long run (induced demand).

There is more and more agreement between transportation professionals that a useful planning tool for such situations is a transportation microsimulation. In such a microsimulation, each traveler is represented as an individual object in the simulation. That makes it straightforward to separate out winners and losers; to "look" for vehicles with cold engines causing excessive pollution; etc. Yet,

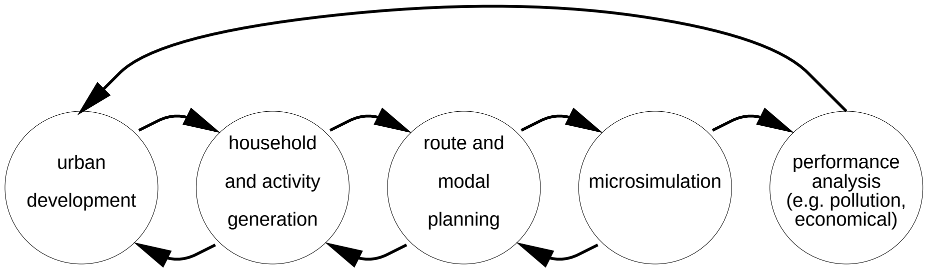

Fig. 1. The TRANSIMS design.

a problem is how do you "drive" such a microsimulation, i.e., how do travelers decide their next move at intersections or at transfer points (train stations etc.)?

The traditional answer to this questions has been "turn counts", i.e. numbers at each intersection which tell you which percentage of vehicles makes left or right turns, respectively. It is fairly clear that this will not work as soon as the infrastructure changes affects people's behavior; e.g. a left turn which has been used heavily before the change is now no longer used much, rendering the turn counts for this particular intersection irrelevant.

It is also fairly obvious that random selection (annealed randomness) often favored by (statistical) physicists has a good chance of not being very helpful in the very non-homogeneous structure of transportation systems.

2 TRANSIMS

It thus seems that the only answer is to drive the traveler objects in the simulation by something which emulates real-world behavior, i.e. by intentions. This, and its realization into a practical computer code, is the core of the TRANSIMS project [3, 4, 5, 6, 7] (see also [8]).

With intentions, one is very soon faced with a consistency problem. Real people presumably plan their trip before they leave their current location (meaning one needs simulated planning), but are open to deviate from the plans for example when the conditions they encounter are much different from what they expected (meaning one needs on-trip planning). Furthermore, the intentions somehow have to be generated in the computer in the first place.

The TRANSIMS approach to this problem is to parcel out the different parts of this process (see Fig. 1):

– Population and activities generation: Stochastically generate a population of individuals for a given geographic area such that the demographics of this generated population matches demographic data. Then, for each individual, generate activities such as work, shopping, social activities, which that individual wants to perform under some scheduling restrictions (e.g. go to work at 9 am; go shopping once a week; etc.).

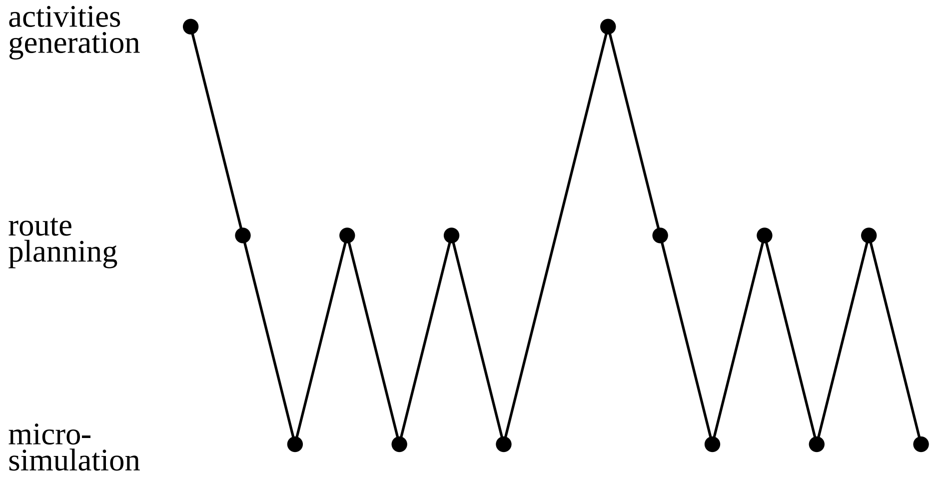

Fig. 2. Visualization of feedback cycles between activity generator, planner, and microsimulation.

– Trip chaining, modal choice and route planning: The activities are combined into trip chains, and the trip is routed on the transportation system, including modal choice
– Microsimulation: The trip plans are executed in a detailed microsimulation of the transportation system
– Analysis: The output of the traffic microsimulation is input into analysis modules, such as air quality, or measures of efficiency analysis

Despite the separation of the modules, it is clear that there are backward causalities. For example, if the microsimulation displays congestion, people will react by choosing different routes. If that does not help, they will re-schedule their activities. If nothing helps, they will maybe relocate to a more convenient location.

This backwards causality is what causes one of the computational challenges. In order to sort out the causal interdependencies, it is necessary to run the microsimulation which results in a new cost function (travel-time) on the infrastructure, then to re-plan according this new cost function, then to run the microsimulation again, etc., until some relaxation criterion is fulfilled. After that, one probably has to re-run the activities generation, adapting activities to what can actually be achieved in the given transportation system. This triggers in return a completely new relaxation cycle between route planner and microsimulation, etc. (see Fig. 2). In consequence, for certain problems the microsimulation may have to be run hundreds of times, thus demanding ultra-high computing speeds.

3 Other computational challenges

There are at least two other computational challenges in the transportation simulation area: systematic analysis, and real time forecasting.

3.1 Simulation as a dynamical system

Reasonable traffic simulations will be Monte Carlo simulations, meaning that different random seeds will produce different system trajectories.[4] That means that only ensembles of multiple simulations actually produce meaningful results. The extent of robustness of the results has to be analyzed; if simulations turn out to be consistently non-robust in certain aspects, then only the distributions of outcomes may be seen as the result. That is, many simulation cycles will have to be run both to find robust and non-robust elements of the simulation and to generate the distributions for the non-robust outcomes.

3.2 Real time forecasting

Above, transportation simulation has been described in the context of a long-term planning problem. Having a twenty years planning horizon, it may be inconvenient but not impossible if the generation of meaningful simulation results for, say, a representative day takes several weeks. Yet, there are other problems such as traveler information (sometimes called ATIS – Advanced Traveler Information Systems) or traffic management (ATMS – Advanced Traffic Management Systems) where at least some of the questions involve real time forecasting. That means that the forecasting procedure has to run faster than real time, i.e. the forecast better be available from the computer before the situation is there in reality.

4 The TRANSIMS microsimulation

Transportation simulation is, as outlined above, an intricate process involving a multitude of different scales, ranging from the representation of the movements of individual travelers to the representation of planning decisions on a weekly scale or more. It is clear from the above remarks that the transportation microsimulation poses one of the major challenges here, both because of the enormous scale of the problem (large regional areas often have 10 million or more travelers simultaneously on route) and because it is in the innermost loop of the relaxation as described above. Yet, also the other modules pose considerable computational loads — for example, generating the 10 million trips which are necessary for a 24 hour run of the Dallas/Fort Worth area takes about 24 hours

[4] Arguably, a deterministically chaotic simulation would do the same when started from slightly varied initial conditions.

on a workstation, and produces several Gigabytes of output. However, reasonably systematic results exist so far only for vehicular traffic microsimulations, which we will concentrate on for the remainder of this paper.

A useful model for traffic microsimulation has to fulfill several fairly obvious criteria: (i) It has to resolve at least individual cars, so that individual plan following can be implemented. (ii) It has to generate reasonable traffic dynamics. (iii) It has to be computationally fast.

TRANSIMS uses a cellular automaton (CA) approach for the traffic microsimulation. The approach is microscopic, it generates realistic traffic dynamics including the characteristic variance structure of certain traffic measurements, and it is computationally fast.

For the CA approach, the road is separated into cells which are either empty, or occupied by exactly one car. The size of the cell, l, is given by the inverse density of a traffic jam: $l = 1/\rho_{jam}$. A good approximation is $l = 7.5$ m. Cars move by jumping from one cell to another. Before they move, they adjust their velocity according to three simple rules: (a) Accelerate if you can up to some maximum velocity v_{max}. (b) Decelerate if you must (i.e. if another vehicle is too close ahead of you). (c) With a certain probability, be slower than that (randomization). See, e.g., [9, 10, 11] for more information. Lane changing and other features are implemented in the same simple, straightforward way, see, e.g., [12, 13, 14].

An update of the system consists of several completely parallel sub-steps (lane changing, intersection dynamics, velocity update and movement). This makes a parallel implementation straightforward: As long as boundary information is exchanged after each sub-step, all sub-steps can be done concurrently.

As a result, the CA approach is computationally very efficient: Due to the simplified dynamics, it already fast on desktop computers, and due to the parallel update, it can be implemented efficiently on parallel computers, yielding linear speed-ups for large enough system sizes.

This approach has been extensively tested on many computer architectures. A simple single-lane circle was implemented using a wide variety of computer architectures (coupled workstations, parallel and/or vectorizing supercomputers) using different algorithms (list-based, bit-coded, etc.). After settling down on a certain implementation technique targeted for certain computer architectures, large realistic road networks were implemented or are under implementation.

5 Implementations of the CA traffic model

For the practical coding, we considered three different approaches: site oriented, particle (= vehicle) oriented, and an intermediate scheme. *Site oriented* directly implements the CA: A street is represented by an array v_j, $j = 1 \dots L$ (L = system size) of integers with values between -1 and v_{max}. -1 means that there is no vehicle at this site, whereas the other values denote a vehicle and its velocity. In contrast, *vehicle 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 vehicle i ($i = 1, \dots, N$).

This is similar to a molecular dynamics algorithm [15], except that vehicles are constrained to integer positions and velocities.

Obviously, the vehicle-oriented approach will always be faster than the site-oriented one for sufficiently low vehicle densities. Yet, for the site-oriented approach single-bit coding (see, e.g., [16]) 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 vehicle. Nevertheless, we found that, on a workstation, the single-bit algorithm is faster than the vehicle-oriented one 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), single-bit coding becomes tiresome. Moreover, for realistic simulations, one needs access to more vehicle information, making a pointer to further vehicle data necessary. Since a pointer typically is a 32-bit number which would have to be moved along in memory with the vehicle in the space-based single-bit approach, single-bit coding is no longer efficient.

With respect to the vehicle oriented approach, the problem is the same as in parallel Molecular Dynamics approaches: How to find the neighbors for interaction, for example for intersection dynamics, lane changing, or car following. Solutions to this are possible but elaborate, and the expected computational speed gain (not more than a factor of about four, see Table 1) did not warrant the additional programming complexity at the current state of the project.

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 [17].

6 Computational speeds on different supercomputers

Table 1 gives an overview of the computational speeds on selected computers. When comparing performance data, it is necessary to give the size of the simulated system. This becomes imperative for parallel computers, since too small systems perform poorly due to the communication overhead. All values of Table 1 have been obtained by simulations of systems of size $L = 10,000$ single-lane km ($1,333,333$ *sites*) with an average traffic density of 13.4 vehicles/km (0.1 veh/site, $134,000$ vehicles in the whole system). This is a system size which is relevant for applications. Moreover, it is a system size small enough to still fit into memory of our single node machines, but which is at the same time large enough to run relatively efficient on our parallel machines. Quantitatively, this means that both the GCel and the CM-5 were operating at 40% efficiency. A larger system size would be even more efficient.

References in the literature often give a "real time limit" as measure of their model's performance, which then is the *extrapolated* system size (or number of vehicles) where simulation is as fast as reality. We found these values practically useless in the area of parallel computing, except when given in conjunction with the system size which has actually been simulated.

In consequence and in order to avoid confusion, our primary table entries are the CPU times we needed on the different machines in order to simulate the system as defined above. For convenience, we also calculated the real time limits in km and in vehicle sec/sec from these values. But it should be kept in mind that, if one really simulates system sizes near 1 million km on the parallel machines, one will find much better real time limits for these system sizes (e.g. 2 million km instead of 900,000 km on the GCel).

Noteworthy features of the table are: (i) The bit-coded CA-algorithm is far superior over the "intermediate" one on the vectorizing machines (NEC SX-3/11 and CM-5), slightly faster on the workstation-based architectures, and slightly slower on the massively parallel Parsytec GCel-3. (ii) All algorithms can take good advantage of the parallelism. (iii) 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. For comparison, the freeway network of Germany is about 60 000 single lane kilometers long (12 000 km $\times$ 2 directions $\times$ 2.5 lanes). We are therefore confident to reach, for a realistic network setup, real time limits of 1,700,000 single lane kilometers ($23,000,000$ veh sec/sec) on 512 nodes of a CM-5, even without using the vector nodes.

Although traffic dynamics is quite different, our computational speeds are comparable to those of Ising models: On the NEC-SX3/11, 0.0025 sec for 10 000 sites correspond to 533 MUPS (Mega-Updates Per Second), which may be compared to 1050 MUPS of a very fast implementation of the (two-dimensional) Ising model on the same computer [18].

7 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 section shows results of systematic measurements.

Fig. 3 gives, for different computers, the best computational speed as a function of system size 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

	s.bit (F77)	particle (F77)	intermed. (C)	netw. (C)
Sparc10	0.33 sec 30 000 km 0.4 e 6 veh	0.15 66 000 km 0.89 e 6 veh	0.71 sec 14 000 km 0.19 e 6 veh	1.14 sec 8 800 km 0.12 e 6 veh
PVM (5× Sp10)	0.07 sec 140 000 km 1.9 e 6 veh		0.15 sec 65 000 km 0.87 e 6 veh	
SX-3/11[(1)] 1 CPN	0.0025 sec 4 000 000 km 53 e 6 veh		0.48 sec 21 000 km 0.28 e 6 veh	
GCel-3 1024 CPNs	0.013 sec 750 000 km 10 e 6 veh	0.0065 sec 1 550 000 km 20.7 e 6 veh	0.011 sec 900 000 km 12 e 6 veh	
iPSC 32 CPNs	0.016 sec 630 000 km 8 e 6 veh		0.038 sec 260 000 km 3.5 e 6 veh	
Paragon 64 CPNs				0.034 sec **290 000 km** **3.9 e 6 veh**
CM-5[(1)] 32 CPNs	0.0077 sec[(2)] 1 300 300 km 17 e 6 veh		0.045 sec[(3)] 220 000 km 2.9 e 6 veh	
CM-5[(1)] 1024 CPNs				[> **1.7 e 6 km**] [> **23 e 6 veh**]

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

Table 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 two entries: the real time limits in km and in vehicle sec/sec.

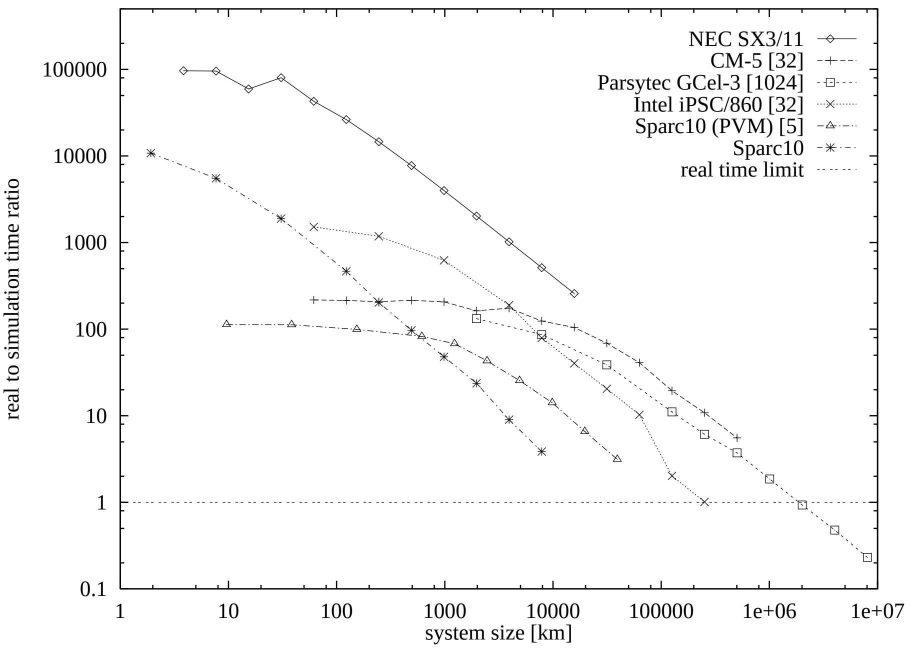

Fig. 3. Comparison between real time and computer time for different system sizes, i.e. the factor r the computer model runs faster than reality. The system sizes where the curves cross $r = 1$ are the so-called real time limits.

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 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 results in an approximate doubling of the computer time, which is visible as a slope of -1 in the double-logarithmic plot of Fig. 3. 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).

The performance gain of the single-bit algorithm against the intermediate one can be seen in Fig. 4. 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

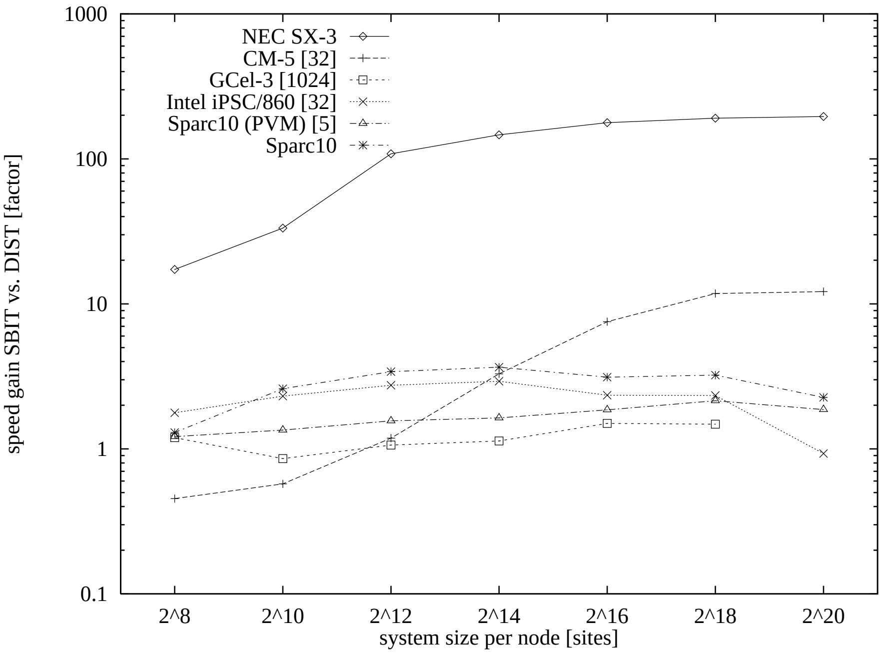

Fig. 4. Computing speed ratio (SBIT divided by DIST) between the two main coding schemes used for our traffic model as a function of system size. Note the low gain on the massively parallel GCel–3 and the huge gains on the vectorizing computers NEC–SX3/11 and the CM–5 (the latter only for large system sizes).

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).

In order to make reliable predictions on scaling behavior, some theory is helpful. Results on a model for the parallel time complexity of the intermediate algorithm can be found in [17].

8 General observations for the implementation of road networks on parallel computers

After the tests with the single-lane model in simple geometries, the CA was enhanced to include multi-lane traffic and to handle complex networks. As mentioned in previous sections, traffic simulations have to be both computationally fast (e.g. at least real-time) and cover a large street network. Due to the limitations in performance of a single workstation, large-scale simulations have to be moved to parallel computers, as soon as the street network size exceeds a certain limit. The straight-forward approach to parallelization is to perform a domain decomposition of the street-network, assigning each sub-network to a computational node (CPN) of the parallel machine. The three most important issues are then: (1) how to cut the network, (2) what state information to exchange along the CPN boundaries, and (3) how to control parallel execution, in order to guarantee a consistent CA update. But, before we address these issues, let us summarize in what aspects the network traffic simulation differs from the original simple geometry.

Network Elements Looking at a graph as the representation of a street network, it is fairly easy to associate edges (streets) as multi-lane CA segment and nodes (intersections) as points where vehicles are transferred from incoming to outgoing segments in an orderly fashion. The actual implementation turns out to be more tedious because on the one hand the effort to administer a network of edges and nodes with individual characteristics is enormous compared to the simple core implementation of the CA. On the other hand, each intersection represents a disruption of the CA grid which is usually regarded as either infinite (periodic boundary conditions) or at least large compared to the boundary length. The algorithms connecting CA segments have to be designed carefully to deliver satisfactory traffic behavior without producing artifacts.

Complex Vehicles and Route-Plans In contrast to earlier investigations of circular traffic in which vehicles were regarded as completely equivalent, each vehicle in the TRANSIMS traffic simulation has an individual route-plan guiding it from its source to its destination, making it unique from any other vehicle in the system. Moreover, due to the requirements regarding the statistical properties which have to be retrieved from the simulation, vehicles also may have to have local memory to store history information of their trips (travel time, e.g. engine temperature, fuel consumption). Therefore, the data which is associated with a vehicle entity is increased many-fold with respect to the few bits (usually coded in on integer) of the original model.

Input/Output A typical setup of CA traffic in a circle looked like this: Generate vehicles with a certain density homogeneously distributed in the system. Start the simulation and let the transients die out. Then, gather aggregated statistics about average velocity and average flux until the run terminates. It should be obvious to the reader that the amount of data produced by the simulation is negligible (maybe a few hundred values) and the input is restricted to a few parameters such as overall density.

The situation looks considerably different in a realistic network simulation: During the preparation phase, each intersection and each segment of the network has to retrieve certain characteristics (e.g. signal phasing, turning prohibitions, length, speed limit, number of lanes) from a data base. After the simulation has started, each route-plan has to be transferred to the source of the route resulting in the instantiating of a vehicle exactly at the departure time-step. As soon as a vehicle reaches its destination, trip data may have to be stored, before the vehicle instance is destroyed. For the Dallas/Fort-

Worth area the number of trips per day can be as high as 3,000,000 resulting in route-plan input files of about 3 Gigabyte.
Most of the questions concerning efficient input/output of the traffic simulation have not been answered yet. The task becomes all the more difficult, since the simulation is only one part in a series of applications (router, traffic simulation, environmental simulation) exchanging large amounts of data. A generalized concept of parallelization has to integrate all these steps of data processing, instead of regarding them stand-alone applications, unnecessarily multiplying data volumes. Insights into these issues will be published in future papers.

8.1 Domain Decomposition

In our opinion, domain decomposition is the most natural approach to parallelization due to the underlying (almost) planar street network. As long as no re-planning (rerouting) is done on-line, the interaction-range of vehicles is restricted to a couple of hundred meters in direction of their travel and even less in the reverse direction. Therefore, for a consistent CA update, only state information of immediate neighborhood is necessary, reducing the boundary area to narrow strip along the borders between neighboring CPN. It should be obvious that the domain decomposition of the network has to be done in such way as to generate the smallest possible boundary area. An easy to implement recursive orthogonal bisection has proven to be (a) computationally efficient, (b) capable to produce satisfactory results, and (c) accessible to a analytical heuristic description, which can be used to predict the parallel efficiency. In our case, the criterion for bisection is defined by load-estimates of the traffic network nodes (e.g. intersection), each of which is computed as the sum Euclidean lengths of all incident edges. This is based upon the assumption that the execution time of the traffic CA strongly depends on the grid size and not so much on the actual vehicle density.

Due the complexity of the intersections, splitting the network at the center of street segments (with one half of the segment updated by either CPN) turned out to be easier than splitting at intersections between nodes and incident segments. The boundary size is then proportional to the total number of segments that had to be split to form the sub-networks. See Figure 5 for an example using 7 CPNs.

Figure 6 depicts the number of inter-CPN segments returned by the recursive orthogonal bisection method and a heuristic formula [19, 20] for three different maps. The similarity between plots and the data points is striking, especially if one considers the different nature of the street networks: (a) the arterials of the Dallas / Fort Worth area, (b) an excerpt of the former, also including local streets, and (c) the Autobahn network of the Federal Republic of Germany.

8.2 Simulation Timing and Boundary Exchange

Due to definition of the CA rule-set, boundary information has to be exchanged between neighboring CPNs before the CA rules can be applied, resulting in

Fig. 5. Domain decomposition example for 7 workstations. On the right is Dallas; for Fort Worth on the left only the major arterials are included. The corresponding microsimulation runs faster than real time.

a basically time-step driven simulation. Depending on the specific coding of the CA rules, more than one sub-time-step may be necessary, each requiring an exchange of boundaries. The basic parallel update scheme works as follows: Each CPN scans its inter-CPN segments and transfers boundary information to its neighbors. Then, it enters a loop waiting for incoming boundaries from its neighbors. As soon as it has received a complete set, it executes a sub-time-step and the cycle is repeated. Note that there is no necessity to trigger a time-step through a master CPN, since each CPN only depends on its neighbors for consistent execution.

As for the length of boundaries, an optimization can be made by taking advantage of specific characteristics of the CA rule set. Usually the boundary that has to be transferred is as large as the *interaction range*[5] of the CA rules, which is currently v_{max}. This would result in encoding and decoding of all vehicle

[5] Actually, the real value that defines the boundary length is the maximum of both *interaction range* and *maximum velocity*, but in a consistent, collision-free CA update

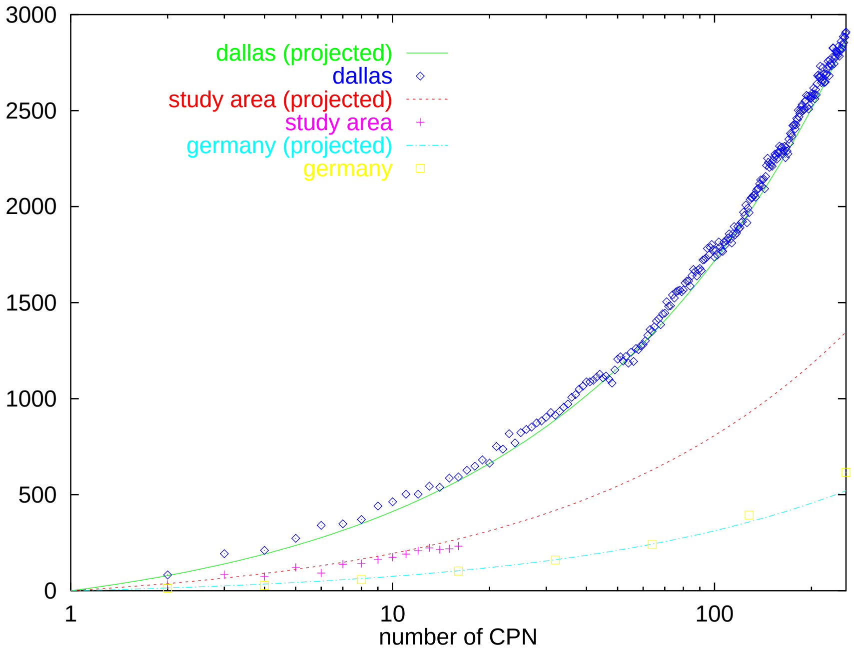

Fig. 6. Number of inter-CPN segments versus number of CPNs.

data that are located within a range of v_{max} sites from a boundary. If the local density is high, that is, the boundary is located within a traffic jam, there may be more than one vehicle per lane. The CA rules, however, only refer to the *immediate* predecessor or successor on each lane, reducing the maximum number of vehicles in a boundary to one per lane. A more detailed description of aspects related boundary exchange can be found in [21].

8.3 Dynamic Load-balancing

Since the simulation is time-step driven, it is the goal to equalize the update times on all CPN. Unfortunately, there may disparities among the CPN, mainly for two reasons: The initial domain decomposition which was based upon a simple estimate, (a) did not include the computational load of the intersection functionality, and (b) did not cover inhomogeneous distribution of traffic during rush hours.

Optional dynamic load balancing can be performed to decrease the load disparities. In one traffic micro simulation [19], the implemented method corresponds to a *local decision, local migration* ($LDLM_S$, see [22]) strategy applied to the network nodes. Incident edges are transferred or split accordingly. When

---

the first is always at least as large as the second. In our current rule-set they happen to be equal.

a part of a local network has to be off-loaded, nodes are sequentially transferred along the boundaries with the node furthest away from the center of the sub-network being selected first. As an optional restriction only those nodes can be selected that maintain one connected component on the CPN.

The flexible data structures which are required to perform dynamic load balancing prove to have another advantage: it is possible to remove or add CPN during the run-time of the simulation by (a) systematically reducing the sub-network on a CPN to zero before removing it, or (b) transferring a single seed node on a newly inserted CPN. In the latter case, the on-going dynamic load-balancing will transfer more and more load the new CPN until it is indistinguishable from the other ones.

8.4 Performance Estimates

We have made a first attempt to deduce an upper bound for the efficiency $e(p)$ of a large scale traffic simulation running on p CPN. It is based upon a set of parameters which can be retrieved from simple measurements. We only cite results here. A more detailed version (including an additional estimate for a two-dimensional communication topology) can be found in [20].

Assuming a time of $T(1)$ required for one time-step on a single-node machine, the necessary input parameters turn out to be:

- the size of the street network (number of edges, number of nodes) resulting in estimates for the number of neighbors $N_n(p)$ (with an average number of $n_n(p) = N_n(p)/p$ per CPN) and the number of boundaries $B(p)$ (with an average number of $b(p)$ per CPN),
- the number of sub-timesteps n_{sub} per time-step causing a relative performance loss for administration overhead $f_{adm}(n_{sub})$ and additional communication volume,
- the average boundary length b_{size} and the boundary message header size b_{header}, the application-level boundary transmission time t_{c1} and transmission latency t_{cl} both given as fractions of $T(1)$,
- the low-level communication bandwidth C_{net} (measured in byte per $T(1)$) of the computer network, and
- the relative load gradient $f_{grad}(p)$ generated by the granularity of the street network.

Using these parameters we define four major contributions to the time spent on one timestep:

- The *raw simulation* fraction mainly represents the traffic simulation itself, although it includes the administrative overhead for multiple sub-time-steps. It is equivalent with the efficiency $e(p)$ of the simulation.
- The *load-gradient* fraction represents the loss of execution time due to the load gradient which builds up throughout the CPN network.
- The *application-level (a-l) communication* fraction represents the time spent on retrieving, coding, transferring, decoding, and storing boundary data.

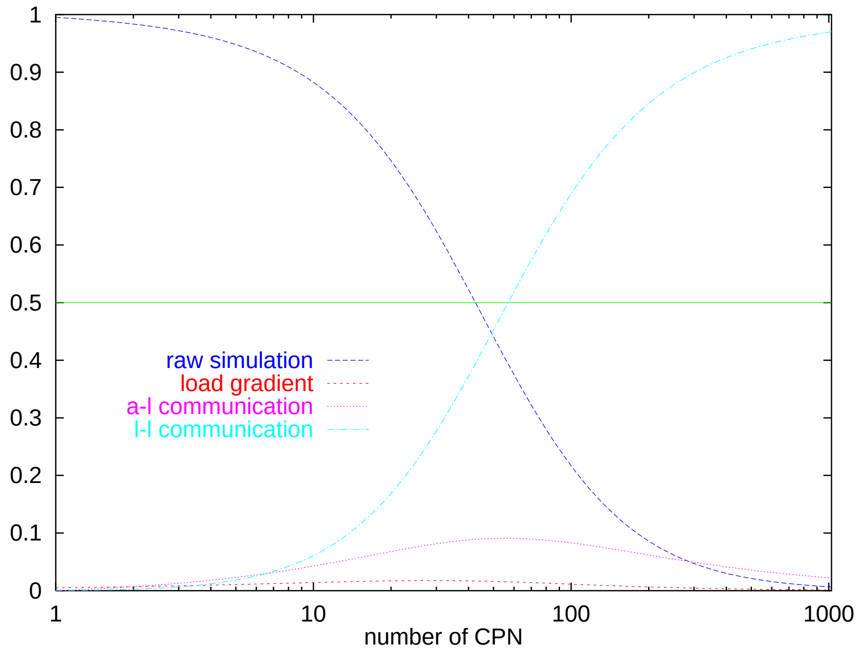

Fig. 7. Sparc-5 Cluster with Ethernet (Bus-Topology)

– Finally, the *low-level (l-l) communication* represents the additional time spent on low-level communication due to the saturation of the underlying communication network.

Figure 7 depicts the estimate for a cluster of Sparc 5 workstations connected by Ethernet. We assumed a relative exponential gradient of 0.01 per layer. Efficiency quickly drops below 0.5 for 30 CPN due to the network saturation by *low-level communication*. Estimates like these can be used to determine the optimal number of CPN to use for a given network size and additional time restrictions.

9 Summary and implications of the computational results

The purely site-based approach is very fast when it uses single-bit coding and runs on machines with vectorizing CPUs. However, in the context of the TRANSIMS project, a simulated vehicle needs additional information associated with it besides position and velocity. That makes a bit-coded approach problematic. In addition, comparable computing speeds can be reached on parallel but non-vectorizing supercomputers.[6]

[6] The single-bit circle implementation on the CM-5 is really fast. However, one will not reach the same computing speeds for realistic networks, which are composed of relatively short links.

For single-lane traffic, the purely vehicle-based approach is very fast for low traffic densities and still a factor of two to three faster than the intermediate approach for realistic traffic densities. For multi-lane traffic, many approaches turn out to be slower than the intermediate approach [23, 24]. And more sophisticated approaches are problematic at the current state of the project since all additional elements of traffic interaction such as intersections, ramps, etc. become more difficult to implement.

For these reasons, TRANSIMS currently uses the "intermediate" approach, intermediate between site-based and vehicle-based. The project is very suited for the use of parallel non-vectorizing supercomputers. Yet, mostly for reasons of client concerns, it currently runs on SparcStations which are coupled by a local area network. Supercomputer implementations are in preparation and will be needed for systematic testing of the dynamics of the simulation.

The reasons why parallel supercomputers are more advantageous compared to coupled workstations lie in the much faster communications network. As a rough number, we can simulate about 50 000 vehicles in real time on a typical desktop CPU. Connecting several of these CPUs via LAN allows simulating larger systems with the same computing speed. However, it is difficult to get faster than 10 times real time with this computational set-up: The communication between workstations is too slow on a LAN.

Dedicated supercomputers are here, in practical terms, about a factor of 10 faster. That means that one can use additional CPNs for enhancing computing speed, not for just making the system larger. A simulation 100 times faster than reality is easily still efficient (see the Intel iPSC example in Fig. 3).

10 Other approaches

The cellular automaton approach is obviously not the only approach to traffic simulation. From a systematic point of view, one has to recognize that one has trade-offs between resolution, fidelity, and scale [25]. Resolution refers to the smallest entities which are resolved in the simulation. For example, a microscopic traffic model resolves individual vehicles. Fidelity refers to the detail with which each entity is modeled. For example, the cellular automaton model has a low fidelity driving dynamics. Scale refers to the system size or temporal scale one can simulate. For example, a model with high resolution and high fidelity will usually run slow, thus only allowing the simulation of a small area and/or short time period.

In consequence, one could try to make the driving dynamics more realistic while preserving the resolution. An obvious option is to use continuous instead of discrete space. Yet, as long as one does not employ an event-driven update (e.g. [26]), one is still stuck with a coarse-grained time step. And all the well-understood techniques from discretized differential equations, i.e. to look for the limit of an infinitely short time step, do not work for traffic, at least not in a straightforward way: In driving, there is a delay between changes in the surroundings and actual changes in the vehicle behavior (given by reaction times, time to move the foot to the break pedal, vehicle inertia, etc.), which is of the order of one second. In consequence, as long as one does not model this delay explicitly but just uses information from the last time-step for the update, time-steps shorter than one second actually lead to *less realistic* traffic dynamics [27, 28, 29]. Moreover, just replacing discrete by continuous space while retaining the one second time-step does not automatically lead to more realistic traffic dynamics. For example, the published fundamental diagram of the PARAMICS project [30] lacks the typical variance structure real world traffic shows under the same circumstances.

PARAMICS is an implementation specially targeted for parallel supercomputers, and has been used on large scale traffic networks [31, 32]. It reaches 250 000 veh sec/sec on a 16k Connection Machine 200 with 512 Floating point units, and 360 000 veh sec/sec on 32 nodes of a Cray T3D [30].

The traditional TRAF/NETSIM [33] uses continuous space and a time-stepped update. A supercomputer implementation on a Cray XMP reached a real time limit of 1000 km [34], which should translate into approximately 13 300 veh sec/sec. Yet, the implementation did not vectorize, so that the simulation does not run much slower on a desktop computer.

Other microscopic models using continuous space and a time-stepped update are, e.g., the Wiedemann model [35], AS [36], MISSION [37, 38], VISSIM [39], THOREAU [40], SISTM [41, 42], and Release 2 INTEGRATION [43, 44]. They have different degrees of fidelity. Most of them have more or less well documented dynamics, but computational speeds are hard to find. Quite in general, it seems that 10 000 veh sec/sec is an upper bound on speeds for these models on a current desktop computer.

For certain questions one may get around a detailed microsimulation. For example, in situations with only very few junctions or intersections, fluid-dynamical approaches may be useful [45, 46, 47, 48, 49], especially for "corridor problems", where all traffic heads for a common destination such as a Central Business District [50, 51]. Sometimes, the traffic dynamics between intersections can be neglected [52].

Schwerdtfeger proposed a model where individual vehicles are moved according to averaged fluid-dynamical rules [53]. DYNASMART [54] and Release 1 of INTEGRATION [55] use similar methods. An early version of DYNASMART reports a computational speed of approx. 600 000 vehicle seconds per second on a CRAY X-MP/24 [56]; since the degree or vecorization is not reported, it is impossible to compare this to other results. Yet, the microsimulation approach, which resolves each individual traveler, is by far the most general and most robust approach. Also, note that INTEGRATION moved to a microscopic approach in Release 2 [44].

11 Some preliminary TRANSIMS results

The TRANSIMS team is currently working on a so-called case study to evaluate and enhance functionality on a practical example and in close collaboration with

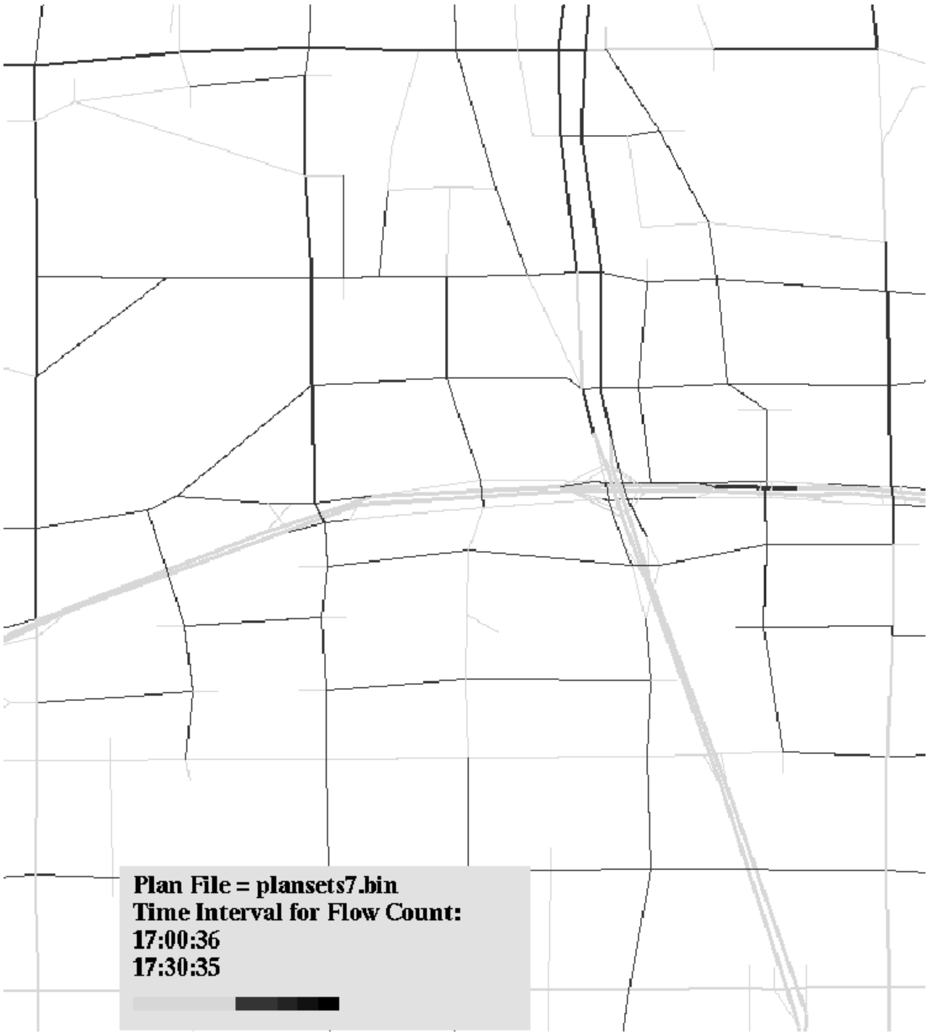

Fig. 8. Superposition of all plans between 17h and 17:30h. On dark roads, demand is higher than capacity, i.e. more vehicles want to go through these roads in that half an hour than what is physically possible. The result has to be congestion on some or all of the roads entering the dark ones.

an MPO (Municipal Planning Organization). A 5 miles $\times$ 5 miles area inside Dallas has been selected as the primary region of interest, but different parts of the project concentrate on different spatial scales.

The focus of this case study is the microsimulation. However, it should be clear from the introduction of this paper that it is difficult to run a meaningful regional microsimulation without plans. (Interestingly, also the inverse is true: Without a microsimulation, it is impossible to evaluate if a certain set of plans

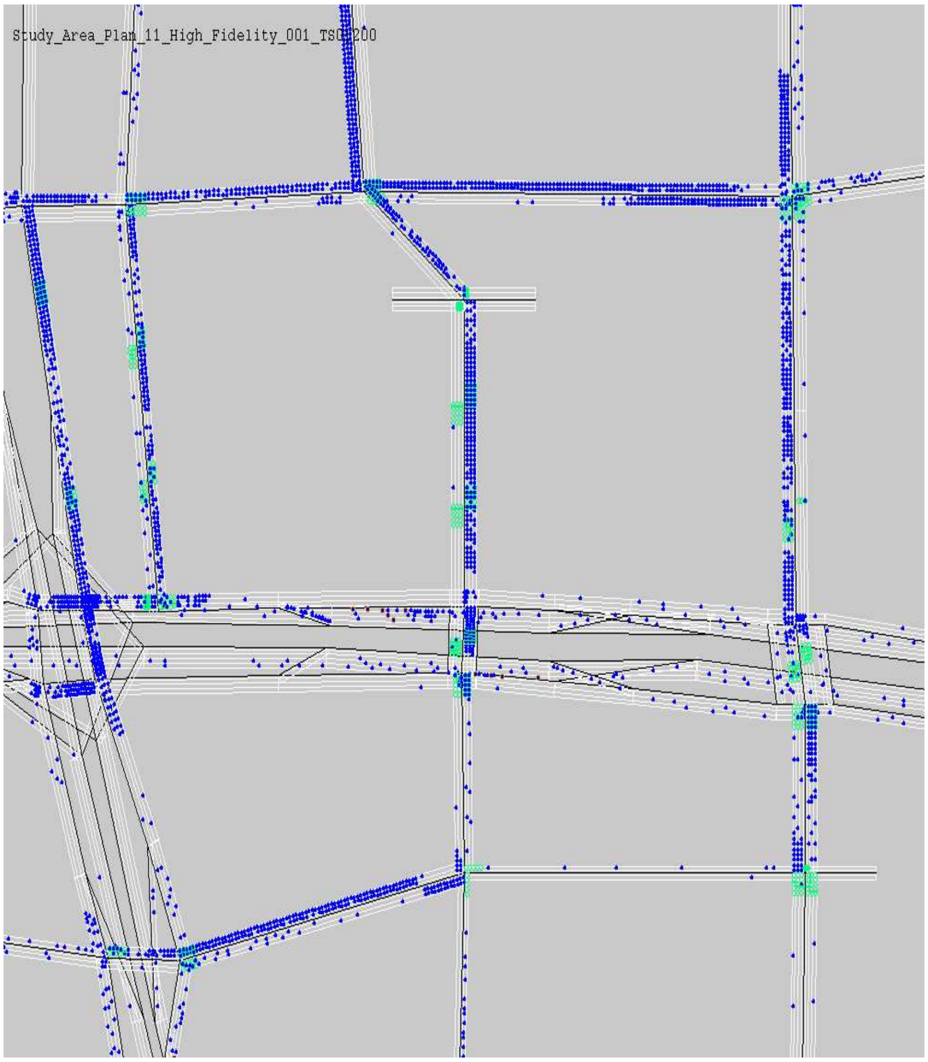

Fig. 9. Simulation of the plans which were underlying Fig. 8. The figure shows only a small part of the simulated area, near the intersection of the two freeways. Note that the freeway coming from the south ("Dallas North Tollway") only extends as a much smaller street to the north. In consequence, congestion builds up on all roads which have traffic heading for that road. Note (see Fig. 8) that demand for this northern part of the Dallas North Tollway was higher than capacity.

makes sense.) In consequence, an interim method to generate plans has been designed. It works as follows:

- NCTCOG (North Central Texas Council Of Governments) provides the TRANSIMS team with time-dependent origin-destination data on a zone basis. This data comes out of more conventional models and is used unchecked.
- TRANSIMS generates individual trips from these aggregate numbers and assigns more specific starting and ending points inside the zones for these trips ("population generation").
- The planner generates route plans for these trips, i.e. for each trip a "reasonable" sequence of links (streets) is generated which connects the starting and the ending points. Only car traffic is assumed (Dallas does not have public transit.)
- Finally, these trips are executed in the microsimulation.

Currently (August 1996), the main problem is to find a reasonable set of plans. In other words: Unrealistic features in the microsimulation are currently dominated by unrealistic features in the plans. Figs. 8 and 9 depict the problem and at the same time visualize some parts of the methodology. Fig. 8 shows a superposition of all plans for the period between 17h and 17:30h. Dark links mean that demand is higher than capacity, i.e. more vehicles want to go through these links in that half hour period than what is physically possible.

Fig. 9 shows a simulation of that set of plans. The figure shows only a small part of the simulated area, near the intersection of the two freeways. Note that the freeway coming from the south ("Dallas North Tollway") only extends as a much smaller street to the north. In consequence, congestion builds up on all roads which have traffic heading for that road. Note (see Fig. 8) that demand for this northern part of the Dallas North Tollway was higher than capacity.

That is, demand which is much higher than capacity for the northern extension of the Dallas North Tollway leads to significant congestion starting on the entrances to that road, and that congestion dominates the dynamics. The reason why this is unreasonable is simple: If this would happen in reality, people would notice and find other ways north. We are currently in the process to investigate relaxation procedures for plansets which emulate this people's behavior and thus should avoid such dominating congestion structures.

Acknowledgements

Much of this paper would not have been possible without the continuous work of all members of the TRANSIMS team. In particular, Dick Beckman has developed the population generation methods. Doug Anson, Myron Stein, and Madhav Marathe have developed the route planning methods. Paula Stretz and the TRANSIMS software team undertook most of the actual implementation. James Dearing and Deborah Kubiceck provided the graphics facilities for the plans and for the microsimulation.

References

1. J. Cohen and F. Kelly. A paradox of congestion in a queueing network. *J. Appl. Probability*, 27:730–734, 1990.
2. M. Williams. Personal communication.
3. L. Smith, R. Beckman, D. Anson, K. Nagel, and M. Williams. Transims: Transportation analysis and simulation system. In *Proc. 5th Nat. Transportation Planning Methods Applications Conference*, Seattle, 1995.
4. J. Morrison and V. Loose. Transims model design criteria as derived from federal legislation. Technical report, Los Alamos National Laboratory, 1995.
5. C. Barrett, K. Berkbigler, L. Smith, V. Loose, R. Beckman, J. Davis, D. Roberts, and M. Williams. An operational description of transims. Technical report, Los Alamos National Laboratory, 1995.
6. L.L. Smith. TRANSIMS Travelogue. *Travel model improvement program newsletter*, 2–5, 1995–1996.
7. TRANSIMS web page. http://www-transims.tsasa.lanl.gov/.
8. R. Schwarzmann. Das EUROTOPP Modell, SCOPE/VIKTORIA-Bericht. Technical report, Institut für Verkehrswesen, TH Karlsruhe, July 1992.
9. K. Nagel. Freeway traffic, cellular automata, and some (self-organizing) criticality. In R.A. de Groot and J. Nadrchal, editors, *Physics Computing '92*, page 419. World Scientific, 1993.
10. K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221, 1992.
11. K. Nagel. Particle hopping models and traffic flow theory. *Phys. Rev. E*, 53(5):4655, 1996.
12. M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 1996. In press.
13. P. Wagner. Traffic simulations using cellular automata: Comparison with reality. In D.E.Wolf, M.Schreckenberg, and A.Bachem, editors, *Traffic and Granular Flow*. World Scientific, Singapore, 1996.
14. P. Wagner, S. Krauss, and C. Gawron. A continuous limit of the Nagel-Schreckenberg model. *Phys. Rev. E*, 1996. Submitted.
15. D.W. Heermann. *Computer simulation methods in theoretical physics*. Springer, Heidelberg, 1986.
16. D. Stauffer. Computer simulations of cellular automata. *J. Phys. A*, 24:909–927, 1991.
17. K. Nagel and A. Schleicher. Microscopic traffic modeling on parallel high performance computers. *Parallel Computing*, 20:125–146, 1994.
18. N. Ito. Non-equilibrium critical relaxation and interface energy of the Ising model. *Physica A*, 1:196, 93.
19. M. Rickert, P. Wagner, and Ch. Gawron. Real-time traffic simulation of the German Autobahn Network. In *Proceedings of the 4th PASA Workshop*, 1996. In press.
20. M. Rickert. Estimating parallel efficiency of large-scale traffic simulations. In preparation, 1996.
21. M. Rickert and P. Wagner. Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. *Int.J.Mod.Phys.C*, 1996. In press.
22. R. Lüling, B. Monien, and F. Ramme. Load balancing in large networks: A comparative study. In *3rd IEEE Symposium On Parallel And Distributed Processing*, pages 686–689, 1991.

23. M.W. Olesen. Personal communication.
24. D. Roberts. Personal communication.
25. C.L. Barrett. Personal communication.
26. T. Blanchard and T. Lake. Conservative spatial simulation. In A. Pave, editor, *European Simulation Multiconference*, page 515. The Society for Computer Simulation, Istanbul, 1993. See also other papers in the same proceedings.
27. C.L. Barrett, S. Eubank, K. Nagel, J. Riordan, and M. Wolinsky. Issues in the representation of traffic using multi-resolution cellular automata. LA-UR 95-2658, Los Alamos, 1995.
28. R. Wiedemann. Personal communication.
29. P. Wagner. Personal communication.
30. G.D.B. Cameron and C.I.D. Duncan. PARAMICS–Parallel microscopic simulation of road traffic. *J.Supercomputing*, 1996. In press.
31. C.I.D. Duncan. PARAMICS wide area microscopic simulation of ATT and traffic management. In J.I. Soliman and D. Roller, editors, *Proceedings of the 28th International Symposium on Automotive Technology and Automation (ISATA)*, page 475. Automotive Automation Ltd, Croydon, England, 1995. Paper No. 95ATS044.
32. PARAMICS Web site. http://www.epcc.ed.ac.uk/epcc-projects/paramics/.
33. U.S.Department of Transportation, Federal Highway Administration. *TRAF user reference guide*, 1992. Publication No. FHWA-RD-92-060.
34. H.S. Mahmassani, R. Jayakrishnan, and R. Herman. Network traffic flow theory: Microscopic simulation experiments on supercomputers. *Transpn. Res. A*, 24A (2):149, 1990.
35. R. Wiedemann. Simulation des Straßenverkehrsflusses. Heft 8, Institut für Verkehrswesen der Universität Karlsruhe, 1974.
36. T. Benz. The microscopic traffic simulator AS (Autobahn Simulator). In A. Pave, editor, *European Simulation Multiconference*, page 486. The Society for Computer Simulation, Istanbul, 1993.
37. R. Wiedemann. Modelling of RTI-elements on multi-lane roads. In *Advanced telematics in road transport, Proceedings of the DRIVE conference*, volume 2. Elsevier, 1991.
38. R. Wiedemann. Simulation des Verkehrsablaufs – Beschreibung des Staus. In *Beiträge zur Theorie des Straßenverkehrs*. Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln, 1995.
39. M. Fellendorf. VISSIM. PTV system GmbH, Pforzheimer Str. 15, 76277 Karlsruhe, Germany.
40. W.P. Niedringhaus and P. Wang. IVHS traffic modeling using parallel computing. In *IEEE - IEE Vehicle Navigation and Information Systems Conference VNIS '93, Ottawa*, 1993.
41. SISTM. Transportation Research Laboratory, Old Wokingham Rd, Crowthorne, Berkshire RG11 6AU.
42. M. McDonald and M. A. Brackstone. Simulation of lane usage characteristics on 3 lane motorways. In *Proceedings of the 27th International Symposium on Automotive Technology and Automation (ISATA)*, 1994.
43. M. Van Aerde, B. Hellinga, M. Baker, and H. Rakha. INTEGRATION: An overview of traffic simulation features. *Transportation Research Records*, in press.
44. M. Van Aerde et al. *INTEGRATION (Release 2) User's Guide*, 1995.
45. H.J. Payne. FREEFLO: A macroscopic simulation model of freeway traffic. *Transportation Research Record 722*, 1979.

46. R.D. Kuehne and R. Beckschulte. Non-linearity and stochastics of unstable traffic flow. In C.F. Daganzo, editor, *Proceedings of 12th Int. Symposium on the Theory of Traffic Flow and Transportation*, page 367. Elsevier, Amsterdam, The Netherlands, 1993.
47. B.S. Kerner and P. Konhäuser. Cluster effect in initially homogenous traffic flow. *Phys. Rev. E*, 48(4):R2335–2338, 1993.
48. D. Helbing. Improved fluid-dynamic model for vehicular traffic. *Phys. Rev. E*, 51(4):3164, 1995.
49. C.F. Daganzo. Requiem for second-order fluid approximations fo traffic flow. *Transpn. Res. B*, 29B(4):277, 1995.
50. M. Kuwahara M and G.F. Newell. Queue evolution on freeways leading to a single core city during the morning peak. In Gartner N H and Wilson N H, editors, *Transportation and traffic theory.* Elsevier, 1987.
51. R.H.M. Emmerink, K.W. Axhausen, P. Nijkamp, and P. Rietveld. Effects of information in road transport networks with recurrent congestion. *Transportation*, 22:21, 1995.
52. H.P. Simão and W.B. Powell. Numerical methods for simulating transient, stochastic queueing networks. *Transportation Science*, 26:296, 1992.
53. T. Schwerdtfeger. *Makroskopisches Simulationmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO).* PhD thesis, TH Karlsruhe, 1987.
54. H.S. Mahmassani, T. Hu, and R. Jayakrishnan. Dynamic traffic assignment and simulation for advanced network informatics (DYNASMART). In N.H. Gartner and G. Improta, editors, *Urban traffic networks: Dynamic flow modeling and control.* Springer, Berlin/New York, 1995.
55. Transportation Systems Research Group, Queens' University and M. Van Aerde and Associates, Ltd. *INTEGRATION: A model for simulating IVHS in integrated traffic networks, User's guide for model version 1.5e*, 1994.
56. R. Jayakrishnan and H.S. Mahmassani. Dynamic simulation-assignment methodology to evaluate in-vehicle information strategies in urban traffic networks. In O. Balci, R.P. Sadowski, and R.E. Nance, editors, *Proceedings of the 1990 Winter Simulation Conference*, 1990.