Distributed intelligence in large scale traffic simulations on parallel computers

Kai Nagel*
Dept. of Computer Science, ETH Zürich
CH-8092 Zürich, Switzerland

August 14, 2002

Abstract

Transportation systems can be seen as displaying meta-intelligence, in the sense that intelligent actors (travelers) conspire to make the system function as a whole. In simulations one can model this by resolving each traveler individually, and giving each traveler rules according to which she/he generates goals and then attempts to achieve them. The system as a whole has no goal of its own except to "function", i.e. to be a metropolitan region where people's lives are good enough so they do not move away.

This paper approaches this question from an extremely pragmatic point – how can a large scale simulation of such a system be implemented on a parallel computer? In particular, we concentrate on Beowulf clusters, which are clusters of regular PCs connected by regular relatively slow local area network. Our working hypothesis is that the imbalance of the computational system –fast CPUs, slow communication– resembles the real system, and that a good simulation of the real system will take advantage of these parallels. The paper focuses both on what is actually implemented and working, and on future plans.

1 Introduction

The real world, we assume, is an example for distributed intelligence. In the archetypical example, the anthill, many agents with limited intelligence –the ants– interact and via this interaction make the whole system –the anthill– function. Similarly, we assume that humans interact to make the whole system –our society– function. We assume that this is achieved by many people making autonomous decisions, i.e. without central control.

The transportation system is a sub-system of this global socio-econo-political system. As we will see, in this system do we not only have agents drive or walk through a networks of roads or walkways, but they also make tactical and strategic decisions, from skipping lunch to relocating their household. That is, the actions of individuals in

*nagel@inf.ethz.ch

the transportation system are strongly coupled to how these individuals live their daily lives, and how they adjust that daily life in reaction to obstacles. In terms of a practical example, a new highway to the subways will often trigger the following reactions: (1) congestion relief; (2) people making additional trips (called *induced* traffic) and/or more people relocating to the subways; and in consequence (3) congestion coming back.

There is an emerging consensus that transportation simulations for planning purposes should consist of the following modules (Fig. 1):

- **Traffic simulation module** – This is where travelers move through the street network by walking, car, bus, train, etc.
- **Modal choice and route generation module** – The travelers in the traffic simulation usually know where they are headed; it is the task of this module to decide which mode they take (walk, bus, car, bicycle, ...) and which route.
- **Activity generation module** – The standard cause why travelers are headed toward a certain destination is that they want to perform a specific activity at that location, for example work, eat, shop, pick someone up, etc. The activity generation module generates synthetic daily plans for the travelers.
- **Life style, housing, land use, freight, etc.** – The above list is not complete; it reflects only the most prominent modules. For example, the whole important issue of freight traffic is completely left out. Also, at the land use/housing level, there will probably be many modules specializing into different aspects.
- **Feedback** – The above modules interact, and the interaction goes in both directions: activities and routes generate congestion, yet (the expectation of) congestion influences activities and routes. This is typically solved via a relaxation method, i.e. modules are run sequentially assuming that the others remain fixed, until the results are consistent.

In addition, there need to be initialization modules, such as the **synthetic population generation module**, which takes census data and generates disaggregated populations of individual people and households. Similarly, it is necessary to generate good default layouts for intersections etc. without always knowing the exact details.

Real-world scenarios often consist of many millions of travelers, and also it seems (without hard evidence) that our multi-agent methods work best on large problems and the corresponding macroscopic questions. For such large problems, parallel computing is an absolute necessity. The first thing to compute in parallel is the traffic micro-simulation – and this is achieved via "standard" domain decomposition, i.e. the geographical region is cut into pieces, and each CPU is responsible for one such piece. Running the other modules in parallel is straightforward as long as the agents do not interact at those levels, as is currently the case for all operational implementations. However, the above relaxation method does not reflect reality – agents do in fact make decisions and change plans *during* travel, and not just before they start. Yet, in a parallel traffic micro-simulation, one cannot have agents go through the cognitive motions of replanning on the same CPU as the traffic simulation is running, since this would

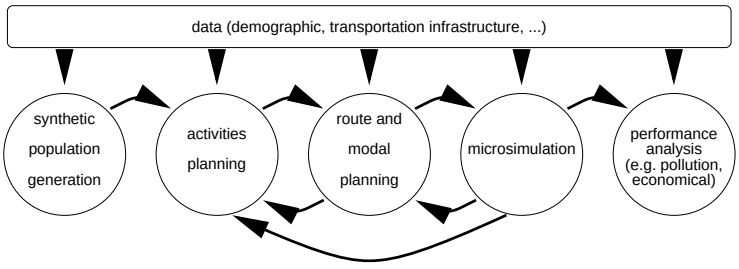

Figure 1: TRANSIMS modules

lead to inefficient load balancing. Thus, the method of choice is to make the intelligence of the travelers external to the micro-simulation – in some sense, to have the traffic micro-simulation represent the "real world" and to have one additional, external, computer for each brain in the simulation.

It should be noted again that this view of distributed intelligence is not oriented towards the solution of any well-defined problem. There is not even a definition of what is meant by intelligence, and if it is each individual agent who is intelligent, or the system as a whole, or both. The only assumption is that the system manages to "function", in the sense that each individual manages to end up with a list of activities which enables him or her to survive, and hopefully to live a good life (in the sense that he or she does not relocate to a different city). This view of distributed intelligence is rather different from a computer science view of distributed intelligence, which takes on the task to achieve a computational speed-up to solve a well-defined problem (e.g. [1]) – although there is certainly overlap, especially in the methods.

We will start with a short discussion of the traditional method (Sec. 2). Next, we present agent-based traffic simulation as an alternative (Sec. 3), and describe the modules mentioned above in more detail. In Sec. 4 we describe how these modules are coupled in order to make the agents adapt and learn. As pointed out above, for large scenarios, parallel computing is a necessity, and it has interesting consequences for the distribution of intelligence in the simulation (Sec. 5). Finally, the state of the art is discussed (Sec. 6), followed by a conclusion.

2 The Four step process and Static assignment

For people with a Complex Systems background it is clear what to do here: Program a version of the real-world spatial substrate, consisting of roads and intersections, and then populate it with agents which follow increasingly sophisticated rules. And in fact, this is what we will describe in the next section. Before we do this, we will however glance at the traditional approach to the problem. This is instructive since it many similarities to a steady-state solution in physics and an equilibrium in economics.

The traditional method of traffic prediction for transportation planning is based on the four step process [2]:

1. **Trip generation:** This module generates, for each traffic zone, the number of trips starting there and the number of trips ending there. This can be done for arbitrary time slices, but is often done for a typical 24-hour weekday.

2. **Trip distribution:** Trip generation results in sources and sinks, but not how they are connected. This is done in the trip distribution module. The result is an **origin-destination matrix**, which has, at row i and column j, the number of trips going from zone i to zone j.

3. **Modal choice:** In this module, the trips are split between the modes of transportation.

4. **Route assignment:** For each trip, a path is found through the network so that no other path is faster. Congestion is taken into account via the link travel time being a function of the trips using that link.

Route assignment has traditionally achieved a lot of attention as a mathematical programming problem. In fact, it can be shown that the solution to a certain version of the above route assignment problem can also be obtained as the solution of a certain non-linear minimization problem. For that reason, many standard methods for non-linear minimization can be used, although certain simplifications are possible because of the structure of the problem [2].

The problem is in fact very similar to a non-linear static network flow problem in physics, where the link "cost" (voltage differential) is given via $U = R\,I$ with a non-constant $R(I)$, and where sources and sinks are given via the result of the trip generation. The only (but important) difference is that in assignment "particles know where they go", meaning that one cannot in general exchange particles as one can, in electrical networks, do with electrons.

Static assignment has many shortcomings. Most importantly, it does not correctly represent dynamic effects such as queue build-up, and it does not have enough microscopic information to do, for example, emission calculations. It also de-couples decisions from individual actors. For example, the only decision available for modal choice is the origin and the destination of the trips; important aspects such as income, car ownership, additional trips during the day, etc. are not used. These latter aspects could however be overcome by a different software design. What cannot be overcome are the shortcomings in the representation of dynamic effects – or, differently stated: when reformulating the dynamics so that it becomes more realistic, most if not all the known mathematical results become invalid, meaning that one loses all the mathematical guidance which has made static route assignment so attractive.

3 Agent-based traffic simulation modules

We will now move on to an agent-based modeling of traffic. In this section, we will describe the fundamental modules of a traffic simulation package, as already presented in the Introduction. As pointed out before, for people with a Complex Systems background it is clear what to do here: Program a version of the real-world spatial substrate,

consisting of roads and intersections, and then populate it with agents which follow increasingly sophisticated rules. We will however also point out one case where the optimal computer-science soluation works better than a complex systems heuristic; and we will describe the standard solutions on which researchers have settled down. In this section, these modules are presented as stand-alone; the issues of module interaction, which results in adaptation and learning, is then treated in the following section.

3.1 Traffic micro-simulation

In order to define a simulation, one first needs to define the spatial substrate. In the case of transportation simulations, this is conventionally a network, consisting of links/edges (roads) and vertices/nodes (intersections).

On these elements, one will have dynamics. There should be pedestrians and cars and buses, etc., and they should drive around according to plausible rules. For car driving alone, the set of absolutely necessary rules is not very large [3], but when including other modes, quite a lot of work is necessary. As a result of this process, one can imagine a virtual reality micro-simulation, such as depicted in Fig. 2.

Sometimes, such an implementation is too much work, and it becomes computationally too slow. It is then possible to replace the virtual reality micro-simulation by something much simpler, which just takes care of moving the travelers through the system and to compute some minimal congestion effects [4, 5].

For the purpose of this paper, the exact nature of the micro-simulation does not matter, as long as it will fulfill some minimum specifications, the most important ones being: (1) vehicles in the traffic micro-simulation follow plans; and (2) the traffic micro-simulation runs computationally fast also on large problems. Although these specifications are currently fulfilled by only very few micro-simulations, let us nevertheless put that problem aside and move on to the other modules.

3.2 Route generation

Having travelers move around randomly is not enough. For example, if a car approaches an intersection, the driver needs to decide the turning direction. A traditional method is to use turn counts, meaning that there is empirical data with the information about what fraction of the traffic goes into which direction. For any kind of transportation planning question, this is not enough information. The most drastic example is the addition of a new road: There would be no information available of how the traffic at the connecting intersection redistributes when the new road becomes connected. One would also assume that turn counts at other intersections change, since some of the traffic would adapt to use the new road.

This means that for transportation planning simulations it is indispensable to know the destinations, and to have routes for each vehicle. In this way, when a new road or a railway connection is added, every traveler can consider to adapt their routing in order to use this new connection. The route generation module of the transportation planning simulation should be multi-modal (i.e. include other modes besides cars), although some of the mode decision is better done in the demand generation module (see next).

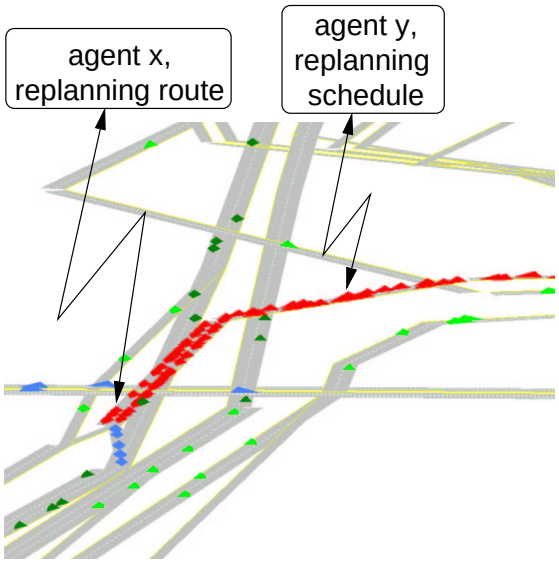

Figure 2: Virtual reality representation of simulated traffic in Portland/Oregon. Including visualization of plans server capability, see Sec. 5.2.

A typical method for route generation is a time-dependent fastest path algorithm. Given a starting time t_0, an origin i and a destination j, and, for each link, information how long it will take to traverse the link when entering at a specific time, this algorithm will compute the fastest path from i to j when starting at time t_0. The time-dependent Dijkstra algorithm, which solves this problem, is with a heap implementation of complexity $M \log N$, where M is the number of links and N is the number of nodes (intersections). This is in fact a very fast algorithm, and it is difficult to construct a heuristic which is significantly faster [6].

3.3 Activity generation

For many questions, having the routes adaptive while the activities remain fixed is not enough. For example, making travel faster usually results in people making more trips. This is called **induced traffic**. Conversely, increasing congestion levels will eventually suppress trips which would otherwise be made, although it is not always clear which trips are suppressed and what congestion level is necessary to have that effect.

In order to deal with these and other effects, one has to make demand generation adaptive to congestion. A recent method for this is activity generation, meaning that, for each individual in the simulation, one generates a list of activities (such as sleeping, eating, working, shopping) plus locations and times (Fig. 3). Since in this method each traveler is treated individually, it is possible to use arbitrary decision rules, which means that arbitrary methods can be investigated. The currently best-accepted methods are based on random utility theory and are called discrete choice models [7]. These

Figure 3: Example of a sequence of activities for a person in Portland/Oregon. From R.J. Beckman.

models lead typically to a form of

$$p_i \propto \exp\big(-\vec{b}\cdot\vec{x}_i\big)$$

for the probability to choose option i, where $\vec{x}_i$ is a vector of attributes. The attributes can refer to the person – e.g. income, gender, marital status, etc. – or to the option – e.g. time spent waiting, time on bus or car, cost of option, etc. They can also be combinations of both; in fact, time on bus or car, say, will depend on where I live. The parameter vector $\vec{b}$ weighs those different attributes; in general the entries of $\vec{b}$ are obtained via estimation from surveys and remain fixed for a typical study, while the entries for $\vec{x}_i$ are taken from the person under consideration and the option under consideration.

As stated above, activity generation needs to be done in conjunction with mode decisions. For example, having a car clearly changes the list of preferable destinations for a given activity, or may even make other activities more desirable. In general, there are no completely clear dividing lines between modules – often, decisions can reasonably be placed in more than one module. It is the task of adaptation and learning (Sec. 4) to consolidate possible inconsistencies between modules.

3.4 Housing, land use, freight, life style, et al

Transportation planning does not stop at activities. For example, making commuting roads faster by increasing capacity usually results in more people moving to the suburbs. That is, housing decisions are closely related to transportation system performance. Similarly, questions of land use (e.g. residential vs. commercial vs. industrial) clearly influence and interact with transportation. Freight traffic needs to be considered. Life style choices (e.g. urban life style, often without car ownership, vs. rural life style, usually with car ownership) need to be considered; such long-term commitments have strong influence on activity selection and modal/route choice.

4 Adaptation and learning in traffic simulation systems

In Sec. 3, we have defined the modules of a multi-agent traffic simulation. What is important is that the agents are not just particles that are moved through the system via some force field, but that they have tactical and strategic goals – tactical for example being the decision to change lanes, strategic for example being to eat three times a day. Sec. 3 has laid out what the corresponding modules do, but not how they interact. This is the topic of this section.

4.1 Day-to-day learning, feedback, and relaxation

The interaction between the modules can lead to logical deadlocks. For example, plans depend on congestion, but congestion depends on plans. A widely accepted method to resolve this is systematic relaxation (e.g. [8]) – that is, make preliminary plans, run the traffic micro-simulation, adjust the plans, run the traffic micro-simulation again, etc., until consistency between modules is reached. Fig. 4 shows an example. The method is similar to a standard relaxation technique in numerical analysis.

Fig. 4 shows an example of the effect of this. The scenario here is that 50 000 travelers, distributed randomly throughout Switzerland, all want to travel to Lugano, which is marked by the circle. The scenario is used as a test case, but it has some resemblance with vacation traffic in Switzerland,

The left figure shows traffic when every driver selects the route which would be fastest on an empty network. The micro-simulation here uses the so-called queue model [9], which is a queueing model with an added link storage constraint. That is, links are characterized by a service rate (capacity), and a maximum number of cars on the link. If the link is full, no more vehicle can enter, causing spill-back. Compared to the original version of Ref [9], our model has an improved intersection dynamics [10]. After the initial routing and the initial micro-simulation, iterations are run as follows:

1. During the micro-simulation, one collects link travel times, averaging over, say, 15 minutes. That is, all vehicles entering a link between, say, 8am and 8:15am, will contribute to the average for that time period.

   Now, for a randomly selected fraction of, say, 10% of the travelers, the old routes are replaced by new routes which are generated based on these averaged link travel times.

2. For a randomly selected fraction of, say, 10% of the travelers, new routes are generated based on the averaged link travel times. As described in Sec. 3.2, this is achieved by running a time-dependent Dijkstra algorithm. The time-dependency is included by using the time-dependent averaged link travel times every time a link is considered.

3. The traffic micro-simulation is run again based on the new set of routes

4. Another 10% of the travelers obtains new routes.

5. Etc., until some kind of convergence criterion is fulfilled.

Fig. 4 right shows the result after 49 such iterations. Quite visibly traffic has spread out over many more different routes.

Such iterated simulations can be treated as very high dimensional time-discrete dynamical systems. A state is the trajectory of the simulation through one day; an iteration is the update from one day (period) to the next (Fig. 5). As such, one can search for properties like fix points, steady state densities, multiple basins of attraction, strange attractors, etc. Typically, one would first analyze the steady state behavior, and then the transients. Under certain conditions the existence of a unique steady state can be proven [11], although for the computationally feasible number of iterations the possible occurrence of "broken ergodicity" [12] needs to be taken into account. Broken ergodicity is the property of a system to be mathematically ergodic but to remain in parts of the phase space for long periods of time.

Fig. 6 shows the relaxation behavior for a scenario in Dallas [13, 14]. The plot shows the sum of all travel times as a function of the iteration number. From this plot and from other observations it seems that here, broken ergodicity is not a problem, and all relaxation methods go to the same state, although with different convergence speeds.

The result is in fact similar to a fixed strategy Nash Equilibrium: for a single run in the relaxed state, it is approximately true that no traveler could improve by changing routes. The players follow essentially a "best reply" dynamics (i.e. find the best answer to yesterday's traffic), and for some systems it can even be proven that this converges to a Nash equilibrium [15]. In our case, we have been able to show for a scenario in Dallas that in a relaxed congested system, the strategy landscape is much flatter than in an uncongested system [16]. This is a signature of a (population-based) mixed strategy Nash Equilibrium.

The relaxed solution is typically better than the initial one, but also worse than some system-optimized solution; in fact, it is relatively easy to construct such scenarios (e.g. [17]). Again, one recognizes that our system finds a "workable" solution, but does not optimize in any way.

4.2 Individualization of knowledge

4.2.1 Classifier System and Agent Database

Knowledge of agents should be private, i.e. each agent should have a different set of knowledge items. For example, people typically only know a relatively small subset of the street network ("mental map"), and they have different knowledge and perception of congestion.

This now completely opens the door for the use of Complex Adaptive Systems methods (e.g. [19]). Each agent has a set of strategies from which to choose, and indicators of past performance for these strategies. The agent normally choses a well-performing strategy. From time to time, the agent choses one of the other strategies, to check if its performance is still bad, or replaces a bad strategy by a new one.

This approach divides the problem into two parts (see also [20]):

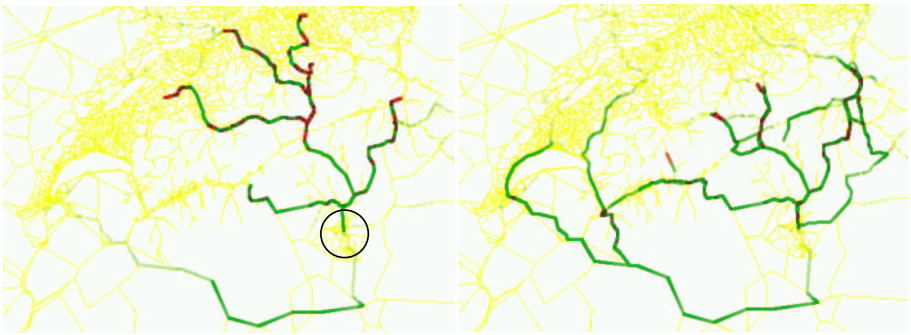

Figure 4: Result of day-to-day learning in a test example. LEFT: Situation at 9:00am in the initial run. RIGHT: Situation at 9:00am in the 49th iteration. Each pixel on the road is a car (by overlapping in the graphics they form the traffic streams); the circle denotes where they are going. Clearly, the system has found a better solution after 49 iterations.

- Plans evaluation. In this phase, plans (or strategies) are evaluated. In our context this means that travelers try out all their different strategies, and the strategies obtain scores. Finally, the agents settle down on the better-performing strategies.

  As usual, the challenge is to balance exploration and exploitation. This is particularly problematic here because of the co-evolution aspect: If too many agents do exploration, then the system performance is not representative of a "normal" performance, and the exploring agents do not learn anything at all. If, however, they explore too little, the system will relax too slowly (cf. "run 4" and "run 5" in Fig. 6).

- Plans generation. In this phase, new plans (or strategies) need to be generated. Since they will be evaluated later, the challenge is to generate a large diversity of strategies, which covers as much as possible the space of possible strategies.

A major advantage of this approach is that it becomes more robust against artifacts of the router: if an implausible route is generated, the simulation as a whole will fall back on a more plausible route generated earlier. Fig. 7 shows an example. The scenario is the same as in Fig. 4; the location is slightly north of the final destination of all trips. We see snapshots of two relaxed scenarios. The left plot was generated with a standard relaxation method as described in the previous section, i.e. where individual travelers have no memory of previous routes and their performance. The right plot in contrast was obtained from a relaxation method which uses *exactly the same router* but which uses an agent data base, i.e. it retains memory of old options. In the left plot, we see that many vehicles are jammed up on the side roads while the freeway is nearly empty, which is clearly implausible; in the right plot, we see that at the same point in time, the side roads are empty while the freeway is just emptying out – as it should be.

The reason for this behavior is that the router miscalculates at which time it expects travelers to be at certain locations – specifically, it expects travelers to be much earlier

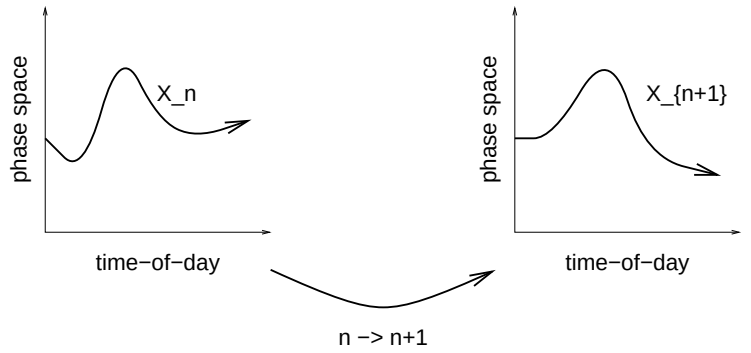

Figure 5: Schematic representation of the mapping generated by the feedback iterations. Traffic evolution as a function of time-of-day can be represented as a trajectory in a high dimensional phase space. Iterations can be seen as mappings of this trajectory into a new one.

at the location shown in the plot. In consequence, the router "thinks" that the freeway is heavily congested and thus suggests the side road as an alternative. Without an agent data base, the method forces the travelers to use this route; with an agent data base, agents discover that it is faster to use the freeway.

This means that the true challenge is not to generate exactly the correct routes, but to generate a set of routes which is a superset of the correct ones [20]. Bad routes will be weeded out via the performance evaluation method. For more details see [21]. Other implementations of partial aspects are [22, 23, 24, 25].

4.2.2 Individual plans storage

The way we have explained it, each individual needs computational memory to store his/her plan or plans. The memory requirements for this are of the order of $O(N_{people} \times N_{trips} \times N_{links} \times N_{options})$, where N_{people} is the number of people in the simulation, N_{trips} is the number of trips a person takes per day, N_{links} is the average number of links between starting point and destination, and $N_{options}$ is the number of options remembered per agent. For example, for a 24-hour simulation of all traffic in Switzerland, we have $N_{people} \sim 7.5$ mio, $N_{trips} \sim 3$, $N_{links} \sim 50$, and $N_{options} \sim 5$, which results in

$$7.5 \cdot 10^6 \text{ persons } \times \ 3 \text{ trips per person } \times \ 50 \text{ links per trip}$$

$$\times \ 5 \text{ options} \times \ 4 \text{ bytes per link } = \ 22.5 \text{ GByte}$$

of storage if we use 4-byte words for storage of integer numbers. Let us call this **agent-oriented plans storage**.

Since this is a large storage requirement, many approaches do not store plans in this way. They store instead the shortest path for each origin-destination combination. This becomes affordable since one can organize this information in trees anchored at

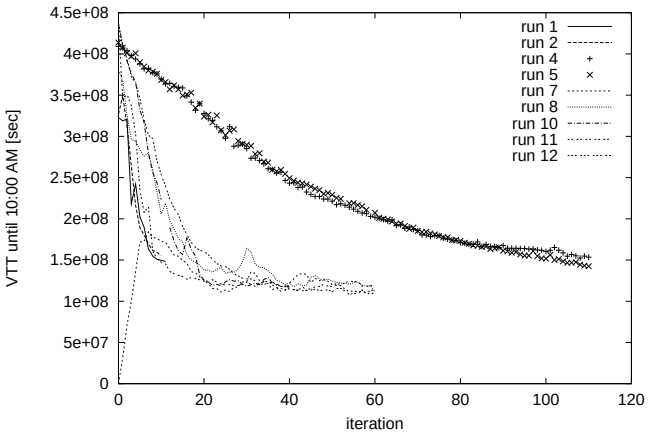

Figure 6: Different relaxation paths in day-to-day replanning. The plot shows the sum of all travel times VTT (Vehicle Time Traveled) as a function of the iteration for different relaxation methods. All methods relax to the same value of VTT. From [18].

each possible destination. Each intersections has a "signpost" which gives, for each destination, the right direction; a plan is thus given by knowing the destination and following the "signs" at each intersection. The memory requirements for this are of the order of $O(N_{nodes} \times N_{destinations} \times N_{options})$, where N_{nodes} is the number of nodes of our network, and $N_{destinations}$ is the number of possible destinations. $N_{options}$ is again the number of options, but note that these are options *per destination*, so different agents traveling to the same destination cannot have more than $N_{options}$ different options between them.

Traditionally, transportation simulations use of the order of 1000 destination zones, and networks with of the order of 10 000 nodes, which results in a memory requirement of

$$1\,000 \text{ destinations } \times \text{ } 10\,000 \text{ nodes } \times \text{ } 5 \text{ options per destination } \times \text{ } 4 \text{ bytes per node}$$

$$= 200 \text{ MByte},$$

considerable less than above. Let us call this **network-oriented plans storage**.

The problem with this second approach is that it explodes with more realistic representations. For example, for our simulations we usually replace the traditional destinations zones by the links, i.e. each of typically 30 000 links is a possible destination. In addition, we need the information time-dependent. If we assume that we have 15-min time slices, this results in a little less than 100 time slices for a full day although some of the information can be recycled [26]. The memory requirements for the second method now become

$$30\,000 \text{ links } \times \text{ } 10\,000 \text{ nodes } \times \text{ } 100 \text{ time slices}$$

$$\times\, 5 \text{ options } \times \text{ } 4 \text{ bytes per entry} \approx 600 \text{ GByte},$$

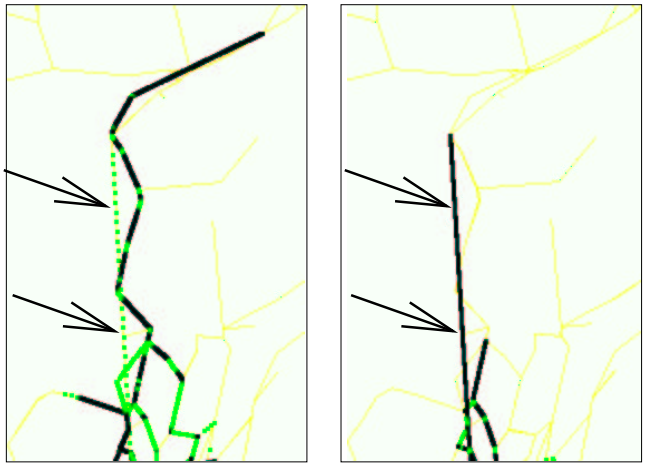

Figure 7: Individualization of plans and interaction with router artifacts. LEFT: All vehicles are re-planned according to the same information; vehicles do not use the freeway (arrrows) although the freeway is empty. As explained in the text, this happens because the router makes erroneous predictions about where a vehicle will be at what time. RIGHT: Vehicles treat routing results as additional options, that is, they can revert to other (previously used) options. As a result, the side road now empty out before the freeway. – The time is 7pm.

already more than for the agent-oriented approach. In contrast, for agent-oriented plans storage, time resolution has no effect. The situation becomes worse with high resolution networks (orders of magnitude more links and nodes), which leaves the agent-oriented approach nearly unaffected while the network-oriented approach becomes impossible. As a side remark, we note that in both cases it is possible to compress plans by a factor of at least 30 [27].

4.3 Within-day re-planning

Day-to-day replanning assumes, in a sense, "dumb" particles. Particles follow routes, but the routes are pre-computed, and once the simulation is started, they cannot be changed, for example to adapt to unexpected congestion and/or a traffic accident. In other words, the strategic part of the intelligence of the agents is external to the micro-simulation. In that sense, such micro-simulations can still be seen as, albeit much more sophisticated, version of the link cost function $c_a(x_a)$ from static assignment, now extended by influences from other links and made dynamic throughout time. And indeed, many dynamic traffic assignment (DTA) systems work exactly in that way (e.g. [8]). In terms of game theory, this means that we only allow unconditional strategies, i.e. strategies which cannot branch during the game depending on the circumstances.

Another way to look at this is to say that one assumes that the emergent properties of the interaction have a "slowly varying dynamics", meaning that one can, for exam-

ple, consider congestion as relatively fixed from one day to the next. This is maybe realistic under some conditions, such as commuter traffic, but clearly not for many other conditions, such as accidents, adaptive traffic management, impulsive behavior, stochastic dynamics in general, etc. It is therefore necessary that agents are adaptive (intelligent) also on short time scales not only with respect to lane changing, but also with respect to routes and activities. It is clear that this can be done in principle, and the importance of it for fast relaxation [28, 18] and for the realistic modeling of certain aspects of human behavior [29, 30] has been pointed out.

4.4 Smart agents and non-predictability

A curious aspect of making the agents "smarter" is that, when it goes beyond a certain point, it may actually *de*grade system performance. More precisely, while average system performance may be unaffected, system variance, and thus unpredictability, invariably goes up. An example is Fig. 8, which shows average system performance in repeated runs as a function of the fraction f of travelers with within-day replanning capability. While average system performance improves with f increasing from zero to 40%, beyond that both average system performance and predictability (variance) of the system performance degrade. In other words, for high levels of within-day replanning capability, the system shows strong variance between uncongested and congested. From a user perspective, this is often not any better than bad average system performance – for example, for a trip to the airport or to the opera, one usually plans according to a worst case travel time. Also, if the system becomes non-predictable, route guidance systems are no longer able to help with efficent system usage. The system "fights back" against efficient utiliziation by reducing predictability.

Results of this type seem to be generic. For example, Kelly reports a scenario where many travelers attempt to simultaneously arrive at downtown for work at 8am [31]. In this case, the mechanism at work is easy to see: If, say, 2000 travelers want to go to downtown, and all roads leading there together have a capacity of 2000 vehicles per hour, then the arrival of the travelers at the downtown location necessarily will be spread out over one hour. Success or failure to be ahead of the crowd will decide if one is early or late, very small differences in the individual average departure time will result in large differences in the individual average arrival time, and because of stochasticity there will be strong fluctuations in the arrival time from day to day even if the departure time remains constant. Ref. [32] reports from a scenario where road pricing is used to push traffic closer towards the system optimum. Also in this case, the improved system performance is accompanied by increased variability. Both results were obtained with day-to-day replanning.

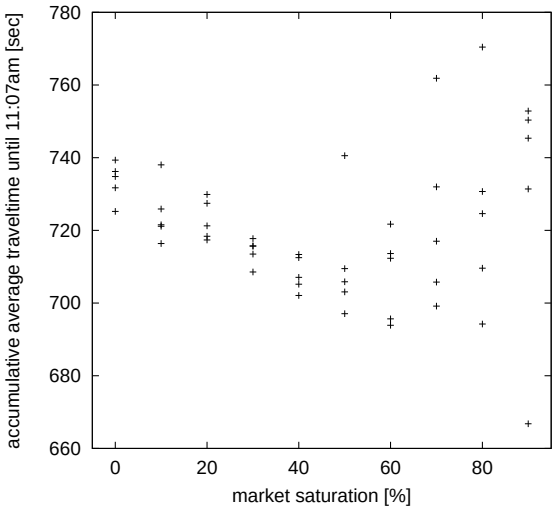

Figure 8: Predictability as function of within-day rerouting capabilities. The result was obtained in the context of a simulation study of route guidance systems. The x-axis shows the fraction of equipped vehicles; the y-axis shows average travel time of all vehicles in the simulation. For each value of market saturation, five different simulations with different random seeds were run. When market saturation increases from zero to 40%, system performance improves. Beyond that, the average system performance, and, more importantly, also the predictability (variance) of the system performance degrade. From [18].

5 Distributed computing and truly distributed intelligence

5.1 Parallel micro-simulations

The most compute-intensive part of current implementations is usually the traffic micro-simulation. A simple calculation gives an approximate number: Assume a 24-hour simulation ($\sim 10^5$ sec) and a one second time step, 10^7 travelers, and a 1 GHz CPU (10^9 CPU-cycles per sec). Further assume that the computation of one time step for each traveler needs 100 CPU-cycles – remember the driving rules (car following, lane changing, protected turns, unprotected turns) and include overhead for route following etc. The result is that such a simulation takes about $(10^5 \times 10^7 \times 10^2)/10^9 = 10^5$ seconds or approximately 1 day on a single CPU. This is indeed approximately correct for a TRANSIMS simulation of a corresponding Switzerland scenario (5 mio travelers; network with 28 622 links); the queue simulation is 10–100 times faster [10].

The simulations can be accelerated by using parallel computers. This becomes indispensable for large applications when including feedback learning as discussed in Sec. 4.1 since this multiplies the computing times by a factor of 50, resulting in

50 days of computing time for the above scenario when using the TRANSIMS micro-simulation. We focus on so-called Beowulf architectures, since they are the most probable ones to be available to prospective users (metropolitan planning organizations; traffic engineering consulting companies; academics). Beowulf clusters consist of regular workstations (such as Pentium PCs running Linux) coupled by regular local area network (such as 100-Mbit Ethernet).

The idea is to divide the simulation area into many pieces, each of which is given to a different CPU. The CPUs communicate e.g. via message passing. In principle, using, say, 100 CPUs should result in a speed-up of 100. In practice, there are many limiting factors coming from the hardware and from the operating system. For traffic micro-simulations, the most important limiting factor is the latency of the Ethernet, which (in an off-the-shelf system without tuning) is of the order of 1 msec [33]. Since each CPU in the average needs to communicate with six other CPUs, this means that each time step needs approx. 6 msec for communication. This limits the speed-up to $1 \text{ sec}/6 \text{ msec} \approx 167$, independent of the number of CPUs that one uses. In practice, "100 times faster than real time" is a good rule of thumb [10, 34]. This domain decomposition approach is similar to a parallel computing approach to "standard" particle dynamics, for example in molecular dynamics [35], with the maybe only distinction that molecular dynamics simulations rarely use a graph instead of regular Cartesian space as spatial substrate.

Unfortunately, in contrast to many other computing aspects, latency does not seem to improve in commodity hardware: is has been virtually unchanged from 10 Mbit Ethernet to 100 Mbit Ethernet to Gbit Ethernet; FDDI is even slower. This has some interesting consequences:

- The above result refers to the speed-up with given system size when using more and more CPUs. Alternatively, one can run larger and larger systems when using more and more CPUs. As is well known, scale-up is much less problematic on parallel computers than speed-up. In consequence, it is possible to run scenarios of virtually arbitrary size 100 times faster than real time.

- Alternatively, one can make the micro-simulations more realistic while still being able to compute 100 times faster than real time.

- It should be noted that parallel supercomputers do not have the same limitation since they employ special purpose hardware for the communication between CPUs. This results in an improvement by a factor of 100 for latency, meaning that for practical scenarios other factors play a more important role.

  While a parallel Beowulf costs of the order of 2000-3000 U.S.-$ per node, a parallel supercomputer is about 20 times more expensive. Since this makes supercomputers irrelevant for the expected users of transportation simulation systems, even when considering the use of a supercomputing center, we have done little research in that direction.

  It is however possible to use more advanced communication hardware for Beowulf clusters, for example Myrinet (www.myri.com). This should improve latency and thus maximum speed-ups by a factor of 10-50.

- Finally, it should be mentioned that, while for 10 Mbit Ethernet the main limiting factor was the hardware, for Gbit Ethernet this is no longer true: Special purpose implementations [36] bring Gbit Ethernet in the range of Myrinet. It is unclear if these improvements will make it into the mainstream.

Alternatively, one can consider other means of speeding up the computation. A possibility is to replace day-to-day replanning by within-day replanning, as discussed in Sec. 4.3. Experiments have shown that this reduces the number of necessary iterations considerably [18]. Possible distributed implementations of this are discussed in Sec. 5.2.

5.2 Truly distributed intelligence

Once the traffic micro-simulation is parallelized, it becomes considerably more difficult to add within-day replanning. As long as one runs everything on a single CPU, it is in principle possible to write one monolithic software package. In such a software, an agent who wants to change plans calls a subroutine to compute a new plan, and during this time the computation of the traffic dynamics is suspended. On a parallel computer, if one traveler on one CPU does this, *all other* CPUs have to suspend the traffic simulation since it is not possible (or very difficult) to have simulated time continue asynchronously (Fig. 9 left).

A better approach is to have the re-planning module on a different CPU. The traveler then sends out the re-planning request to that CPU, and the traffic simulation keeps going (Figs. 2 and 9 right). Eventually, the re-planning will be finished, and its result will be sent to the simulated traveler, who picks it up and starts acting on it. An experimental implementation of this using UDP (User Datagram Protocol) for communication shows that it is possible to transmit up to 100 000 requests per second per CPU [24], which is far above any number that is relevant for practical applications. This demonstrates that such a design is feasible and efficient.

Some readers may have noticed that success of the re-planning operation is not guaranteed. For example, the new plan may say to make a turn at a specific intersection, and by the time the new plan reaches the traveler, she/he may have driven past that point. Such situations are however not unusual in real life – how often does one recognize that a different decision some time ago would have been beneficial. Thus, in our view the key to success for large scale applications it to not fight asynchronous effects but to use them to advantage. For example, once it is accepted that such messages can arrive late, it is also not a problem to not have them arrive at all, which greatly simplifies message passing.

An additional advantage of such a distributed design is that the implementation of a separate "mental map" (Sec. 4.2) for each individual traveler does not run into memory or CPU-time problems. Specific route guidance services can be simulated in a similar way. Also, non-local interaction between travelers becomes a matter of direct interaction between the corresponding "strategic" CPUs, without involving the rest of the computational engine. This occurs for example for ride sharing, or when family members re-organize the kindergarten pick-up when plans have changed during the day, and will necessitate complicated negotiations between agents. However, neither

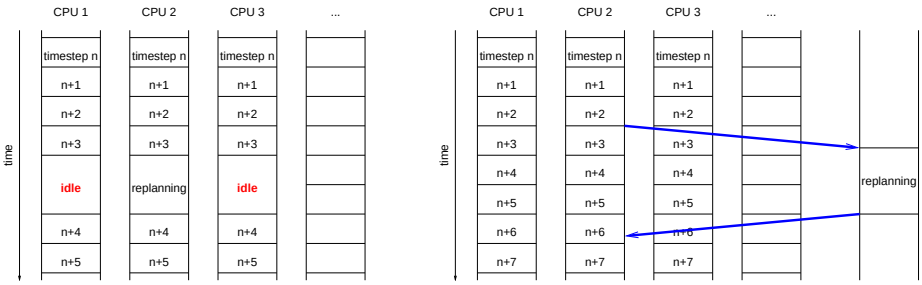

Figure 9: Parallel implementation of within-day replanning. LEFT: Implementation as subroutine of parallel traffic simulation. RIGHT: Implementation via separate plans server.

the models nor the computational methods for this are developed.

This design is similar to many robot designs, where the robots are autonomous on short time scales (tactical level) while they are connected via wireless communication to a more powerful computer for more difficult and more long-term time scales (strategic level); see, e.g., Ref. [37] for robot soccer. Also, the human body is organized along these lines – for example, in ball catching, it seems that the brain does an approximate pre-"computation" of the movements of the hands, while the hands themselves (and autonomously) perform the fine-tuning of the movements as soon as the ball touches them and haptic information is available [38]. This approach is necessitated by the relatively slow message passing time between brain and hands, which is of the order of 1/10 sec, which is much too slow to directly react to haptic information [39].

That is, in summary we have a design where there is some kind of "real world dynamics" (the traffic simulation), which keeps going at its own pace. Agents can make strategic decisions, which may take time, but the world around them will keep going, meaning that they will have to continue driving, or deliberately park the car. As pointed out, such an architecture is very well supported by current distributed computers, although the actual implementation still needs to be done.

6 State of the art

No simulation package currently integrates all the aspects that are discussed. TRANSIMS [40] comes from the transportation planning side and is maybe the most advanced in terms of using many of the concepts. The TRANSIMS research program is reaching completion in 2002, with a full-scale simulation of a scenario in Portland/Oregon, with a network of 200 000 links and several million travelers. TRANSIMS uses an agent-based approach to how the travelers make decision, including an agent data base and a selector which selects agents for replanning. TRANSIMS does however not implement any of the "within-day replanning" aspects discussed in Sec. 4.3. We ourselves are in the process of using TRANSIMS for a full-scale simulation of all of Switzerland [41]. We also have prototypes for the "truly distributed

intelligence" as discussed above [24], and for genetic algorithm usage for plans generation [42]. DYNAMIT [43] and DYNASMART [44], originally started as transportation simulation tools for the evaluation of ITS (Intelligent Transportation System) Technology, also advance into the area of transportation planning by the addition of the demand generation modules. METROPOLIS [45] is a package designed to replace static assignment by a simulation-based but very simple dynamic approach. It allows the user to specify arbitrary link-cost functions but in its current version is does not allow the queue build-up which is important for congested systems. The strength of METROPOLIS lies in the self-consistent computation of departure time choice. Very few projects use individual plans. Instead, they use shortest-path trees as described in Sec. 4.2. A collection of articles about regional transportation simulation models can be found in [46].

Thus, for real world implementations, there is still a long way to go until the agent-based approach is truly implemented, let alone tested. As an example of the few existing comparisons to real world data, Fig. 10 shows such a comparison for a Portland (Oregon) scenario done within the TRANSIMS project. The compared data are hourly flows, i.e. the number of cars crossing certain measurement locations during an hour. For both plots, the x-coordinate of a point is given by the field data value, while the y-coordinate is given by the model result. In consequence, the deviation from the diagonal is a measure of how much field data and simulation result disagree. Each point denotes a different measurement location. The left plot shows results of our simulation, while the right plot shows results from a model run done by the Portland transportation planning authority using more traditional technology. The result says that agent-based simulations in transportation currently are, in terms of the quality of the result, comparable to the more traditional technology; this statement can be quantified [47]. When interpreting this result, one should consider that our result was preliminary, with a much simplified micro-simulation and a much simplified demand generation, while on the other hand the Portland transportation authority has a reputation for excellent modeling work. For further details, see [47].

7 Conclusion

Socio-economic systems are systems with distributed intelligence: Each agent makes decisions on her/his own, which together makes the system function as a whole. In contrast to many other distributed intelligence approaches, there is however no overarching problem to solve or to optimize – it is enough if the system "works".

The transportation system is a part of the socio-economic system, and it functions according to the same principles: travelers make autonomous decisions, and somehow the system conspires to "work", i.e. fulfill the transportation needs of each individual. New agent-based simulation approaches to transportation use the same principles: the simulation is composed of agents, and besides driving they also make decisions on the strategic level such as planning their daily activities and choosing their mode and route. Although it is not yet universally accepted and very few implementations exist, it seems that in future these simulations will have an agent database where each agent collects several strategies and corresponding performance knowledge, and in conse-

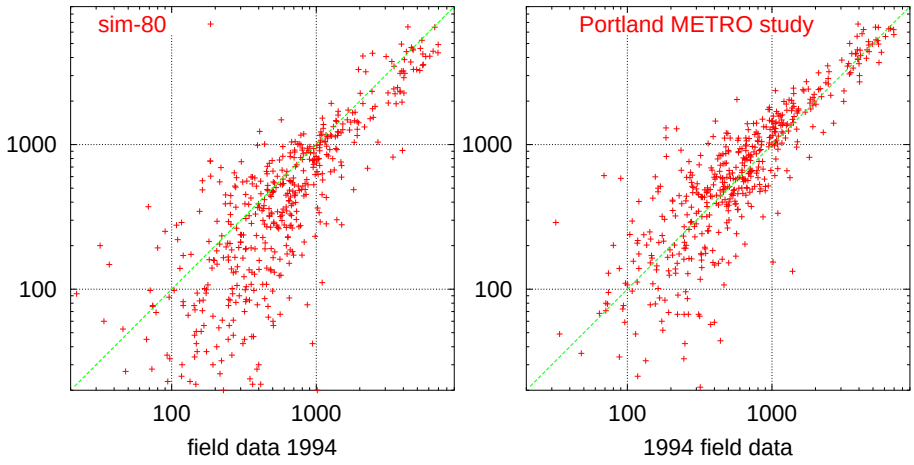

Figure 10: Comparison between field data and model results. LEFT: Our method. RIGHT: Portland transportation planning authority

quence, each agent will have only a partial and individualized view of the situation. In addition, simulation systems will allow for with-day replanning, i.e. that agents are able to change their plans spontaneously and not just over night.

For large scenarios, parallel computing is a necessity. The arguably cleanest way to do this is to have two structures: (1) A parallel traffic micro-simulation, where a regular daily traffic dynamics unfolds according to the tics of some clock. Agents in this simulation are autonomous on the tactical level. (2) Distributed modules which compute strategic decisions of the agents. These modules will compute and update strategies while the dynamics keeps unfolding. Once they have settled down on a strategy, this is communicated to the agent in the micro-simulation, which will implement it if it is still consistent with what has happened in the meantime.

Acknowledgments

Los Alamos National Laboratory makes the TRANSIMS software available to academic institutions for a small charge.

The Swiss Federal Administration provides the input data for the Switzerland studies.

I thank my collaborators N. Cetin and B. Raney for providing valuable input and their most recent results, and we all thank our collegues at the Insitute for Transportation, Traffic, Highway- and Railway- Engineering, in particular K. Axhausen and M. Vrtic, for considerable help with the “real world” aspects of the Switzerland project.

References

[1] G. Weiss, editor. *Multiagent Systems. A modern approach to distributed artificial intelligence*. The MIT Press, 1999.

[2] Y. Sheffi. *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1985.

[3] K. Nagel, P. Stretz, M. Pieck, S. Leckey, R. Donnelly, and C. L. Barrett. TRANSIMS traffic flow characteristics. Los Alamos Unclassified Report (LA-UR) 97-3530, Los Alamos National Laboratory, see transims.tsasa.lanl.gov, 1997.

[4] C. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(3):393–407, 1998.

[5] P. M. Simon and K. Nagel. Simple queueing model applied to the city of Portland. *International Journal of Modern Physics C*, 10(5):941–960, 1999.

[6] R. R. Jacob, M. V. Marathe, and K. Nagel. A computational study of routing algorithms for realistic transportation networks. *ACM Journal of Experimental Algorithms*, 4(1999es, Article No. 6), 1999.

[7] M. Ben-Akiva and S. R. Lerman. *Discrete choice analysis*. The MIT Press, Cambridge, MA, 1985.

[8] J.A. Bottom. *Consistent anticipatory route guidance*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[9] C. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. In *International Journal of Modern Physics C* [4], pages 393–407.

[10] N. Cetin and K. Nagel, submitted. See www.inf.ethz.ch/personal/nagel/papers.

[11] E. Cascetta and C. Cantarella. A day-to-day and within day dynamic stochastic assignment model. *Transportation Research A*, 25A(5):277–291, 1991.

[12] R. Palmer. Broken ergodicity. In D. L. Stein, editor, *Lectures in the Sciences of Complexity*, volume I of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 275–300. Addison-Wesley, 1989.

[13] M. Rickert and K. Nagel. Experiences with a simplified microsimulation for the Dallas/Fort Worth area. *International Journal of Modern Physics C*, 8(3):483–504, 1997.

[14] R.J. Beckman et al. TRANSIMS–Release 1.0 – The Dallas-Fort Worth case study. Los Alamos Unclassified Report (LA-UR) 97-4502, Los Alamos National Laboratory, see transims.tsasa.lanl.gov, 1997.

[15] J. Hofbauer and K. Sigmund. *Evolutionary games and replicator dynamics*. Cambridge University Press, 1998.

[16] T. Kelly and K. Nagel. Relaxation criteria for iterated traffic simulations. *International Journal of Modern Physics C*, 9(1):113–132, 1998.

[17] H.S. Mahmassani and S. Peeta. Network performance under system optimal and user equilibrium assignments: Implications for advanced traveler information systems. *Transportation Research Record*, 1408:83–93, 1993.

[18] M. Rickert. *Traffic simulation on distributed memory computers*. PhD thesis, University of Cologne, Germany, 1998. See www.zpr.uni-koeln.de/~mr/dissertation.

[19] J.D. Holland. *Adaptation in Natural and Artificial Systems*. Bradford Books, 1992. Reprint edition.

[20] M. Ben-Akiva. Route choice models. Presented at the Workshop on "Human Behaviour and Traffic Networks", Bonn, December 2001.

[21] B. Raney and K. Nagel. Iterative route planning for modular transportation simulation. In *Swiss Transport Research Conference*, Monte Verita, Switzerland, March 2002. See www.strc.ch.

[22] H. Unger. An approach using neural networks for the control of the behaviour of autonomous individuals. In A. Tentner, editor, *High Performance Computing 1998*, pages 98–103. The Society for Computer Simulation International, 1998.

[23] H. Unger. *Modellierung des Verhaltens autonomer Verkehrsteilnehmer in einer variablen staedtischen Umgebung*. PhD thesis, TU Berlin, 2002.

[24] Chr. Gloor. Modelling of autonomous agents in a realistic road network (in German). Diplomarbeit, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, 2001.

[25] S. Weinmann. *Simulation of spatial learning mechanisms*. PhD thesis, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, in preparation.

[26] I. Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Records*, 1645:170–175, 1998.

[27] B. W. Bush. Personal communication.

[28] J. Esser. *Simulation von Stadtverkehr auf der Basis zellularer Automaten*. PhD thesis, University of Duisburg, Germany, 1998.

[29] K.W. Axhausen. A simultaneous simulation of activity chains. In P.M. Jones, editor, *New Approaches in Dynamic and Activity-based Approaches to Travel Analysis*, pages 206–225. Avebury, Aldershot, 1990.

[30] S. T. Doherty and K. W. Axhausen. The developement of a unified modelling framework for the household activity-travel scheduling process. In *Verkehr und Mobilität*, number 66 in Stadt Region Land. Institut für Stadtbauwesen, Technical University, Aachen, Germany, 1998.

[31] T. Kelly. Driver strategy and traffic system performance. *Physica A*, 235:407, 1997.

[32] K. Nagel and S. Rasmussen. Traffic at the edge of chaos. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 222–235. MIT Press, Cambridge, MA, 1994.

[33] K. Nagel and M. Rickert. Parallel implementation of the TRANSIMS microsimulation. *Parallel Computing*, 27(12):1611–1639, 2001.

[34] P. Gonnet. A thread-based distributed traffic micro-simulation. Term project, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, 2001.

[35] D.M. Beazley, P.S. Lomdahl, N. Gronbech-Jensen, R. Giles, and P. Tamayo. Parallel algorithms for short-range molecular dynamics. In D. Stauffer, editor, *Annual reviews of computational physics III*, pages 119–176. World Scientific, 1995.

[36] http://pdswww.rwcp.or.jp/, since 1993.

[37] J.H. (editor) Kim. Special issue about the first micro-robot world cup soccer tournament, MIROSOT. *Robotics and Autonomous Systems*, 21(2):137–205, 1997.

[38] D. Sternad. personal communication.

[39] J.D. Rothwell. *Control of Human Voluntary Movement*. Chapman and Hall, 1994.

[40] TRANSIMS. TRansportation ANalysis and SIMulation System, since 1992. Los Alamos National Laboratory, Los Alamos, NM. See transims.tsasa.lanl.gov.

[41] A. Voellmy, M. Vrtic, B. Raney, K. Axhausen, and K. Nagel. Status of a TRANSIMS implementation for Switzerland. Networks and Spatial Economics, forthcoming. See www.inf.ethz.ch/˜nagel/papers.

[42] D. Charypar. Genetic algorithms for activity planning. Term project, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, in progress. See www.inf.ethz.ch/ nagel/papers.

[43] DYNAMIT, since 1999. Massachusetts Institute of Technology, Cambridge, Massachusetts. See its.mit.edu. Also see dynamictrafficassignment.org.

[44] See www.dynasmart.com. Also see dynamictrafficassignment.org.

[45] A. de Palma and F. Marchal. Real case applications of the fully dynamic METROPOLIS tool-box: an advocacy for large-scale mesoscopic transportation systems. *Networks and Spatial Economics*, 2002.

[46] K. Nagel and P. Wagner (editors). Special issue on regional transportation simulations. *Networks and Spatial Economics*, forthcoming. See www.inf.ethz.ch/~nagel/papers.

[47] J. Esser and K. Nagel. Iterative demand generation for transportation simulations. In D. Hensher and J. King, editors, *The Leading Edge of Travel Behavior Research*, pages 659–681. Pergamon, 2001.