Transportation planning II: Complex systems applications for transportation planning

Kai Nagel and Bryan Raney
Institute for Scientific Computing, ETH Zürich,
8092 Zürich, Switzerland

May 31, 2003

1 Introduction

A possible goal for transportation and regional planning is to design the transportation or regional systems so that the people who use them will be happy. Yet, happiness is difficult to define in a way which is useful for engineering. For most people, a certain level of economic performance is probably a pre-requisite for happiness, and thus economic indicators are important factors in the design. Other important indicators might include noise, pollution, safety, access to a variety of destinations, or even something as intangible as beauty. These indicators may have different impacts on the happiness of different users of the above-mentioned systems, so design of these systems should somehow account for individuality of the users.

The traditional quantitative method for transportation planning is the *four-step process* (e.g. Sheffi, 1985). The four steps are trip generation, trip distribution, mode choice, and route assignment. In the first two steps, the origins (e.g. residential areas) and destinations (e.g. workplaces) are determined and then connected. The typical result of these steps is an origin-destination matrix, which gives, for each zone in the simulated region, the number of trips to each other zone in a given time slice. The last two steps then determine which mode and which path these trips take.

The main disadvantage of this technology is that all relation to individual people is severed in the process. This has consequences both on the level of modeling and on the level of analysis. For example, on the level of modeling one would expect people's income or distance to a bus stop to be important for mode choice, but the four step method does not take these into account. Similarly, the fact that individual travelers' characteristics are not available makes any analysis based on individual people's characteristics impossible.

An alternative, which will be discussed in this paper, is to maintain the individuality of travelers throughout the modeling process. Doing this is now possible with ever more powerful computers, and the new computational tools that have emerged with them. The computational tools that are particularly relevant with respect to this paper are the methods of *Complex Systems*, which includes for example Cellular Automata, Genetic and other Evolutionary Algorithms, or Multi-Agent Simulation.

The general idea of such an approach is easy to explain: In a multi-agent transportation simulation, travelers are represented as individual "agents" who make independent decisions about their actions. As will be explained later, these decisions range from short-term decisions about acceleration or lane changing all the way to decisions about the planning of daily activities, and later versions of such a modeling system are bound to also include long-term decisions about, say, residential choice.

Note that such an approach still allows planners to extract more conventional aggregated quantities, such as, for example, the fraction of travelers who chose public transit. In contrast to more traditional modeling approaches, in the agent-based system this number will be generated by looking at each traveler individually, and carefully weighting all her/his different options, taking into account for example her/his income and her/his distance to the next bus stop. This is important since it is expected that such details play an important role with respect to choice behavior. In consequence, the aggregated number will transmit the same information as before, but it is now based on a much more microscopic evaluation than before. The same argument applies to all kinds of transportation system analysis. As further examples, consider an analysis of who is affected by noise or air pollution (children!), who is contributing to a traffic jam (do they have an alternative?), or who should pay for public transit (which frees up roads for car users).

This paper introduces the most important aspects of such a multi-agent simulation system for transportation planning, and in fact for all kinds of mobility simulations. Such a simulation system consists of many modules which need to work together. On the highest level, one can distinguish the following classes of modules: First, the movement simulation itself, which moves the agents through the system, and which is concerned with the physical aspects of the system such as volume exclusion (two travelers/cars cannot be at the same location at the same time) or limits on acceleration/braking. This is treated in Sec. 2. Second, tactical/strategic modules model higher-level decision-making such as route choice, generation of daily plans, or residential choice. This is described in Sec. 3. Finally, there needs to be a learning framework, which models how how agents can learn to improve their decisions as the simulation progresses (Sec. 4). After the discussion of these three conceptual pieces of the simulation system, Sec. 5 describes a scenario of real-world rush hour traffic in Switzerland that was executed by our multi-agent transportation simulation system, and results from that simulation. Section 6 presents some issues with computational performance of multi-agent simulation modules. The paper is concluded by a summary.

2 Movement Simulation

At the heart of any simulation that is concerned with the mobility of people – or other agents – is some module that simulates the movement of those people. This can be easiest imagined as some kind of virtual reality environment, which looks “just like reality,” and where any interested party can be immersed in the scene and observe the situation. As we know from movies and from computer games, it is now possible and sometimes desirable to have a near-realistic version of the real world in the computer, if one is willing to accept a relatively small scenario and the necessary effort. On the other hand, for realistic traffic simulations with several millions of travelers, the amount of knowledge, coding, and computational power for such an environment is currently out of reach, so that simplifications need to be made. The degree of simplification depends on the goal, and can sometimes be radical. The traditional static assignment technique (see, e.g. Sheffi, 1985), which may be known to some readers, can be seen as such a radical simplification, which goes so far that all dynamics and all notions of individual travelers are simplified away. In this section, we will concentrate on so-called microscopic simulation ("micro-simulation") techniques which leave at least these two aspects intact.

2.1 Cellular Automata Techniques

2.1.1 Links

A very intuitive technique for micro-simulations are so-called *cellular automata (CA)* (von Neumann, 1966; Wolfram, 1986). Many variations of CA techniques exist; the most basic CA have coarse-grained discrete space (cells), coarse-grained state space, coarse-grained discrete time, and parallel update. For example, for CA pedestrian simulations (Meyer-König et al., 2001), the size of a cell would be such that it can contain at most one pedestrian (coarse-grained discrete space), it would always contain either exactly zero or exactly one pedestrian (two states), the time step would be such that during one time step a pedestrian typically moves from one cell to the next, and all pedestrians would compute their movement simultaneously based on information at time t (parallel update).

Let us now look at traffic simulations instead. The basic entity of a traffic simulation is the road link. For the CA technique, it is cut into cells, say of length $7.5\ m$, which can contain at most one car each, and there is one array of cells for each lane (Fig. 1(a)). Movement is performed by jumping from one cell to another, where the new cell is determined by a set of "driving rules." A good update time step is $1\ sec$ (justified by reaction time arguments). Taking this together means, for example, that a jump of 5 cells in a time step models a speed of $5 \times 7.5\ m/sec = 135\ km/h$.

Driving rules of traffic on a link consist of two parts: car following, and lane changing. Typical rules for car following are: deterministic speed calculation, randomization, and movement. The *deterministic speed calculation* rule first computes a new speed for each car based on its current speed and closeness to the car in front of it. An example for such a rule is

$$v_{safe,t} = \min[v_{t-1} + 1, v_{max}, g_t] \ ,$$

where $v+1$ models acceleration, v_{max} is a speed limit, and g is the gap, i.e. the number of empty spaces to the vehicle ahead. See Fig. 1(a) for an illustration of gap. Next, the *randomization* rule introduces a probability p_{noise} for each car so that instead of following the above speed deterministically, it may instead drive one velocity level more slowly:

$$v'_t = \begin{cases} \max[0, v_{safe,t} - 1] & \text{with probability } p_{noise} \\ v_{safe,t} & \text{else} \end{cases} \ .$$

In the *movement* rule, each particle is moved forward:

$$x_{t+1} = x_t + v'_t \ .$$

To illustrate the above rules, Fig. 1(a) shows traffic moving to the right. Using a typical p_{noise} value of 0.2, the leftmost vehicle accelerates to velocity 2 with probability 0.8 and stays at velocity 1 with probability 0.2. The middle vehicle slows down to velocity 1 with probability 0.8 and to velocity 0 with probability 0.2. The right most vehicle accelerates to velocity 3 with probability 0.8 and stays at velocity 2 with probability 0.2. Velocities are in "cells per time step." All vehicles are then moved according to their velocities using the movement rule described above.

Fig. 2 shows, in a so-called space-time diagram for traffic, the emergence of a traffic jam. Lines show configurations of a segment of road in second-by-second time steps, with time increasing in the downward direction; cars drive from left to right. Integer numbers denote the velocities of the cars. For example, a vehicle with speed “3” will move three sites (dots) forward. Via this mechanism, one can follow the movement of vehicles from left to right, as indicated by an example trajectory.

Typical rules for *lane changing* (see Fig. 1(b)) include a reason to change lanes and a safety criterion for changing lanes. First, there needs to be a reason why a vehicle wants to change lanes, for example that the other lane is faster, or that it needs to get into a certain lane in order to make an intended turn at the end of the link. A possible rule to model the first is “check if the (forward) gap in the other lane is larger than the gap in the current lane.” Second, the vehicle needs to check that there is really enough space in the destination lane. A possible rule is that the forward gap in the other lane needs to be larger than v_t, and the backward gap in the other lane needs to be larger than the velocity of the oncoming vehicle.

Figure 1(b) illustrates the lane changing rules. Only lane changes to the left lane are considered. In situation I, the leftmost vehicle on the bottom lane will change to the left because the forward gap on its own lane, 1, is smaller than its velocity, 3; the forward gap in the other lane, 10, is larger than the gap on its own lane, 1; the forward gap in the target lane is large enough; and the backward gap is large enough. In situation II, the second vehicle from the right on the right lane will not accept a lane change because the gap backwards on the target lane is not sufficient.

Due to the complexity of the dynamics, it is inconvenient to do both lane changing and car following in one parallel update. It is however possible, and convenient for parallel computing, to do the update in two completely parallel sub-timesteps: First, all lane changes are executed in parallel based on information from time t, resulting in intermediate information, at time $t+\frac{1}{2}$. Then, all car following rules are executed in parallel, based on information from time $t+\frac{1}{2}$, which results in information for time $t+1$.

2.1.2 Intersections

The typical modeling substrate for transportation is the network, which is composed of links and nodes that model roads and intersections, respectively. Since the full simulation of an intersection is rather complicated, for large scale applications one attempts to reduce the complexity by resolving all conflicts at the entering of the intersection. Under such a simplification, it is still possible to model things like left turns against oncoming traffic, but the decision is made before the vehicle enters the intersection, not inside the intersection. In consequence, complications inside the intersection, such as the interesting dynamics of a large Parisian roundabout, are not modeled by this approach. Nevertheless, such a simplified approach is quite useful for large scale applications.

Given this, there are only two remaining cases for traffic: Protected and unprotected turns. A protected turn means that a signal schedule takes care of possible conflicts, and thus the simulation does not have to worry about them. In this case, it is enough if the simulation makes vehicles stop when a signal is red, and lets them go when a signal is green.

In unprotected turns, conflicts are resolved by the legal rules, not by signals. For example, a vehicle making a left turn needs to give priority to all oncoming vehicles (Fig. 1(c)). This entails that one needs, for each turning direction, to encode which other oncoming lanes have the priority. Once this is done, a vehicle that intends to do a certain movement only needs to check if there is a vehicle approaching on any of these interfering lanes. If so, the vehicle under consideration has to stop, otherwise it can proceed. As said before, once the vehicle has entered the intersection, other vehicles are no longer regarded. Figure 1(c) shows an example of a left turn against oncoming traffic. The turn is accepted because on all three oncoming lanes, the gap is larger or equal to three times the first oncoming vehicle's velocity.

2.2 Queue Model

Sometimes, even the relatively simple CA rules are computationally too slow. Another factor of ten in the execution speed can often be gained by simplifying the dynamics even more. In the so-called queue model (Gazis, 1974; Gawron, 1998), links have no internal structure any more, they can be considered as vehicle reservoirs. These reservoirs have a storage constraint, N_{max}, which varies from link to link, and is based on the physical attributes of the link, such as length and number of lanes. Once the storage constraint is exhausted, no more vehicles can enter the link until some other vehicles have left the link.

Vehicles can only leave the link if their destination link has space, if they have spent the time it takes to traverse the link, and if the flow capacity constraint is fulfilled. The flow capacity constraint is a maximum rate with which vehicles can leave the link.

Except for the storage constraint and its consequences, this is just a regular M/M/1 queuing model. The storage constraint does however necessitate one adaptation of the model: Since destination links can be full, there is now competition for space on the destination link. A good option is to give that space randomly to incoming links, with a bias toward incoming links with higher capacity (**?**).

2.3 2D Space

As discussed above, the transportation system is abstracted as a network made up of links and nodes. Sometimes, this kind of abstraction is not realistic enough. Examples for this are the modeling of traffic across a complicated intersection, or the modeling of pedestrians. In these cases, a full two-dimensional representation is more useful. This full two-dimensional representation can for example be achieved by two-dimensional cellular automata. The main problem with 2D CA is that there is no good way to model off-axis movement. Although average off-diagonal directions can be maintained by using corresponding probabilities, it is difficult to control the variability of the resulting trajectories. Similarly, speed is difficult to control for off-axis movement: A velocity of one cell per time step along the axis corresponds, for example, to $\frac{1}{\sqrt{2}}$ cells per time step in the diagonal direction.

An alternative is a standard particle approach, where each particle has continuous position and velocity, and particle movements are computed by determining the interactions between every pair of particles. The general difficulty with such approaches is that for each particle one has to go through all other particles just to find out if they are near enough so that some interaction takes place. It is thus necessary to maintain some kind of data structure that avoids having to go through all other particles. A similar problem concerns the interaction of particles with the environment: In a naive implementation, for all objects it needs to be checked if there is an interaction.

In such situations, hybrid approaches are useful. An example of a hybrid approach is to give the particles continuous positions and velocities, but to do all other computations on a grid. In addition, grid cells maintain information about which particles are inside them. With this, if an interaction range is limited, one only needs to search the cells that are within the interaction range. And interactions with the environment, as long as they are static, can be pre-computed entirely at the start of the simulation. Fig. 3 shows an example of this.

3 Agent Strategies (Plans)

As pointed out in the introduction, a simulation for particle movement is not enough to solve questions of transportation planning – a simulation of the particles' tactical and strategic behavior, such as destination choice and route selection, is also necessary. This is where this line of work deviates from standard physics and veers into the social sciences.

In fact, this last statement is only partially correct. As one will see, our work concentrates on simulations with large numbers of particles (agents), thus emphasizing statistical effects of agent interaction over detailed individual mental models. The intuition behind that is that the real world system is governed by a large number of constraints (road capacity, location of residential/commercial areas, demographic structure of the population) and that those constraints play a dominant role in the behavior of the real system. Whether this intuition is indeed correct is an open question.

Typical modules for strategy generation in a multi-agent transportation simulation are: synthetic population generation, activities generation, mode and route choice. These will be described in the following.

3.1 Synthetic Population Generation

A typical first step in giving strategies to the particles is to individualize them by generating a synthetic population. This is done by taking a model of the real environment, and populating it with synthetic agents which have the correct demographic characteristics. For example, within census blocks there will be the correct number of males or females, and the synthetic population will have the correct age structure. What is typically missing in publicly available census data is the correlation structure. For example, the number of males/females and the number of people below/above age 30 may be known (the "marginals"), but not how many males are below/above age 30. Such correlation structure is typically given for a small sample of the population, for which much more information is made available. In return, the geographic locations of these sample households are much less precise. A technique is thus needed to merge these two pieces of information, i.e. to generate for each census block complete tables where the marginals still match the census, and the correlation structure approximates the sample data. A possible technique for this is Iterative Proportional Fitting (IPF; Beckman et al. (1996)). The population is then located along the streets in the census block. None of our current results use the synthetic population generation, but its application is planned in 2003.

3.2 Activity Generation

For each member of the synthetic population, daily activity plans are then generated. This consists of three sub-problems: The first sub-problem is *activity pattern generation*. An activity pattern describes the sequence of activities that an individual performs during a day. Typical examples are "home-work-shop-home" or "home-university-leisure-home."

Next comes *activity location generation*. Activities need to be executed at certain locations. Algorithms are necessary to generate these locations. This will use land use data in order to have information at which locations certain activities can be performed.

The last sub-problem is *activity scheduling*. This is the computation of when the activity chain is started and how long each activity takes. It should take into account an estimate of expected travel times between activities at different locations.

Although methods are known for the above problems (e.g. Bowman, 1998), few projects seem to have implemented this for large scale real-world applications. Most of the emphasis seems to be on the generation of activity patterns, which can be generated from travel diaries which are available for samples of the population. Activity location generation is a problem because of the large number of possible options; some heuristic reduction of the search space will probably be necessary. In contrast, activity scheduling seems to be computationally feasible; yet, there seems to be no universal agreement on cost/utility functions which are necessary to evaluate and compare different schedules (K. Axhausen, personal communication).

The activity generation problem illustrates that one should consider concepts both from computational search and from psychology. Search methods from computer science may result in optimal solutions, or may show that an exact solution of the problem is very difficult. In contrast, real people do not solve these problems optimally, but use heuristics that lead to "good" solutions. For example, Doherty and Axhausen (1998) point out that in reality people solve parts of the activity scheduling problem on different time scales. Approximate work hours are usually known many days in advance, and regular tasks such as bringing children to the kindergarten follow closely behind. More irregular activities such as shopping or leisure activities are planned on a shorter time scale, but usually at least a day in advance. Finally, spontaneous decisions or unexpected events (such as a sick child) can override the schedule at the last minute.

A computational method that lies in between human heuristics and exact algorithmic search are Genetic Algorithms (GAs; Goldberg (1989)). It is in fact possible to use GAs for activity generation (Charypar, 2002). These methods are however far away from being calibrated and ready to use for real-world applications.

Note that activities can be generated for vacation tourists as well as for everyday workers. An activity chain for a tourist would maybe read "hotel – be at mountain X and enjoy view for 10 minutes – be at mountain restaurant Y and have coffee for an hour – hotel." In this case, the travel connecting these locations would not just be considered as a necessary nuisance, but it would contribute to positive utility. The overall concept remains similar.

None of our current real-world results use activity generation, but its application is planned in 2003.

3.3 Mode and Route Choice

Activities at different locations are connected via travel. Individuals make a mode choice (walk, bus, bicycle, car, ...) and a route choice. 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 on how long it will take to traverse the link when entering at a specific time, this algorithm computes the fastest path from i to j when starting at time t_0. The time-dependent Dijkstra algorithm, which solves this problem, is a very fast algorithm, and it is difficult to construct a heuristic which is significantly faster (Jacob et al., 1999).

Mode choice can either be seen as part of the activity generation process (after all, activity selection depends critically on the means of transportation that one selects), or it can be seen as the problem to select a good mode given the chain of activities. In the first situation, it will have to be solved along with activity generation. In the second situation, it is in fact possible to extend the time-dependent Dijkstra algorithm such that mode choice is included (Barrett et al., 2000). Another possibility is to just generate routes for several different modes and then to select between them using a random utility model.

4 Agent Learning

When considering the modules of Sec. 3, it becomes quickly clear that there is no unique direction of causality. The mode choice follows from the activities, but the availability of certain modes influences activity selection. Similarly, congestion information is needed when calculating activity schedules, which in turn influence activity patterns and routes, and thus congestion.

A possible way to solve this problem is to use systematic relaxation. This is where all agents make some choices (plans, or strategies), these plans are executed in a traffic simulation, some agents revise their plans, they are again executed, etc., until some kind of stopping criterion is fulfilled. This approach is widely accepted for the route assignment part, where travelers switch to new routes until they cannot find another route that makes them better off than the route they already have. This is just the definition of a game theoretic Nash Equilibrium.

For activities, it is less clear where such a relaxation algorithm will lead, and if the result makes sense. In fact, there are several issues with such iterations: The first issue is that the Nash Equilibrium (NE) is a normative state; that is, it is claimed that the system somehow reaches that state, and once there it stays at it. This means that it does not matter how the computation reaches this state, any measure to speed up the computation is allowed, and in consequence the feedback iterations do not need to have any relation to real-world human learning. Second, there is no guaranty that the NE is unique. If it is not unique, then any attractive NEs can be reached by the iterative procedure. The result then depends on the initial conditions. Finally, if real people typically do not operate at a Nash Equilibrium, then the method is no longer correct. In that case, it becomes necessary to model human learning directly, i.e. the human adaptation from one day or week or year to the next. In this case, it also becomes necessary to define the speed of human learning. As with multiple Nash Equilibria, initial conditions matter here.

It is possible to write agent-based simulations such that they allow the modeling of both approaches: fast relaxation toward NE, or realistic human learning. An important aspect of such a flexible method is to make the implementation truly agent-based. By this we mean that there are true individuals in the simulation, with home address, demographic characteristics, etc. Each individual has plans for activities, modes, and routes. This is already different from many implementations, where the routes are given implicitly via the destination and are thus not explicitly part of the agents' strategies.

From here, one can make progress by using methods from Evolutionary Game Theory (e.g. Hofbauer and Sigmund, 1998), from Complex Adaptive Systems (e.g. Stein and others, since 1988), Distributed Artificial Intelligence (e.g. Ferber, 1999), and Machine Learning (e.g. Russel and Norvig, 1995). The agent is considered as an intelligent entity that is capable of collecting information about its environment, and of using that information to come up with better and better solutions. Initially, the agent will just attempt to come up with a good plan for itself, but it will have to react to other agents' behavior, especially to congestion. So far, this is a non-cooperative co-evolution problem. Eventually, however, one will have to make the agents negotiate with each other; for example, there will be household tasks which only one member of the household needs to do.

Finally, there is a difference between so-called day-to-day learning, and within-day learning. In the former, agents execute their daily plan without modification, and only "over night" can consider using a different plan. Within-day learning implies that agents can modify their plans during the day. The latter is more realistic since some of our decisions are made on time scales much shorter than a day (Doherty and Axhausen, 1998). It is however also conceptually less clear since day-to-day learning has relations to evolutionary game theory. In our current implementation, we use day-to-day learning.

5 A Real-World Scenario

While the previous sections have discussed general design issues for multi-agent transportation simulations, this section will describe results of a particular scenario. Since this is work in progress, the implementation does not use all the modules described earlier. It will however provide some intuition how multi-agent models can be used in this context, and how they perform when compared to more conventional approaches. The scenario consists of approximately 1 million agents traveling throughout Switzerland during an average week-day morning rush hour, from 6:00 AM to 9:00 AM.

5.1 The "Switzerland" Scenario

The street network that is used was originally developed for the Swiss regional planning authority (Bundesamt für Raumentwicklung), and covered Switzerland. It was extended with the major European transit corridors for a railway-related study (Vrtic et al., 1999). The network has the fairly typical number of 10 564 nodes and 28 622 links. Also fairly typical, the major attributes on these links are type, length, speed, and capacity constraint. From Sec. 2.2, one can see that this is enough information for the queue simulation.

Our starting point for demand generation is a 24-hour origin-destination matrix from the Swiss regional planning authority (Bundesamt für Raumentwicklung). The original 24-hour matrix is converted into 24 one-hour matrices using a three step heuristic (Vrtic and Axhausen, 2002). These hourly matrices are then disaggregated into individual trips. That is, we generate individual trips such that summing up the trips would again result in the given OD matrix. The starting time for each trip is randomly selected between the starting and the ending time of the validity of the OD matrix.

This leads to a list of approximately 5 million trips, or about 1 million trips between 6:00 AM and 9:00 AM. Since the origin-destination matrices are given on an hourly basis, these trips reflect the daily dynamics. Intra-zonal trips are not included in those matrices, as by tradition.

5.2 Simulation models

The above scenario was fed into two different models: First, into a VISUM (PTV, www.ptv.de) assignment which is a relatively standard assignment (Sheffi, 1985) except that it is dynamic on an hourly basis (Vrtic and Axhausen, 2002), and second into an agent-based micro-simulation based on the principles described earlier. For the agent-based simulation, neither a synthetic population nor activities were used, since demand was given by the OD matrix. An initial set of routes was generated using a fastest path algorithm based on free speeds given by the network data. These routes were then run through the queue micro-simulation (Sec. 2.2). The simulation collected link travel times in 15-min time bins. These time-dependent link travel times formed the basis for the next route calculation, which was done for 10% of the travelers. These routes were then merged with the pre-existing routes by the other travelers, and that set of routes was run again through the simulation. These iterations are repeated until the system is relaxed, which takes about 50 iterations.

As a major difference to many other implementations of similar systems, our system remembers *all* routes ever tried by an agent. That is, new routes keep coming in at a rate of 10% new routes per iteration, but even the remaining 90% of agents are adaptive in every iteration: They keep scores (trip times) T_i for each route, and select a route with probability

$$p_i \propto e^{-\beta T_i} \ ,$$

where β tunes the amount of randomness. The use of the agent database makes the software system considerably more robust against artifacts of the router module: "Bad" routes are tried out once by the agent and subsequently ignored. Artifacts of the router module may not just be due to implementation errors, but also due to inherent limitations, such as the one caused by the binning of the travel times (Raney and Nagel, 2003). The agent database also moves the implementation closer to a true agent-interpretation of the traveler, meaning that in such an implementation the agent is an autonomous entity in the system capable of adaptation and learning.

5.3 Results

Figure 5 shows a result of the Switzerland morning rush-hour scenario. This figure is after 50 iterations of the queue micro-simulation, using the agent database. We used as input the origin-destination matrices described in Sect. 5.1, but only the three one-hour matrices between 6:00 AM and 9:00 AM. This means any travelers beginning their trips outside this region of time were not modeled. As one would expect, there is more traffic near the cities than in the country. Jams are nearly exclusively found in or near Zurich (near the top). This is barely visible in Fig. 5, but can be verified by zooming in (possible with the electronic version of this paper; see also sim.inf.ethz.ch/papers/ch). As of now, it is unclear if this is a consequence of a higher imbalance between supply and demand than in other Swiss cities, or a consequence of a special sensitivity of the queue simulation to large congested networks.

Fig. 6(a) shows a comparison between the simulation output of Fig. 5 and field data taken at counting stations throughout Switzerland (see Sec. 5.1 and Bundesamt für Strassen, 2000). The dotted lines, drawn above and below the central diagonal line, outline a region where the simulation data falls within 50% and 200% of the field data. We consider this an acceptable region at this stage since results from traditional assignment models that we are aware of are no better than this (Fig. 6(b); see also Esser and Nagel, 2001). Fig. 6(b) shows a comparison between the traffic volumes obtained using the VISUM assignment (Vrtic and Axhausen, 2002) against the same field data.

Visually one would conclude that the simulation results are at least as good as the VISUM assignment results. Table 1 confirms this quantitatively. Mean absolute bias is $\langle q_{sim} - q_{field} \rangle$, mean absolute error is $\langle |q_{sim} - q_{field}| \rangle$, mean relative bias is $\langle (q_{sim} - q_{field})/q_{field} \rangle$, mean relative error is $\langle |q_{sim} - q_{field}|/q_{field} \rangle$, where $\langle . \rangle$ means that the values are averaged over all links where field results are available. For example, the "mean relative bias" numbers mean that the simulation underestimates flows by about 5%, whereas the VISUM assignment overestimates them by 16%. The average relative error of the simulation is 25%, compared to 30% for the VISUM assignment. These numbers state that the simulation result is better than the VISUM assignment result. Also, the simulation results are better than what we obtained with a recent (somewhat similar) simulation study in Portland/Oregon (Esser and Nagel, 2001); conversely, the assignment values in Portland were better than the ones obtained here.

What makes our result even stronger is the following aspect: The OD matrices were actually modified by a VISUM module to make the assignment result match the counts data as well as possible. These OD matrices were then fed into the simulation, without further adaptation. It is surprising that even under these conditions, which seem very advantageous for the VISUM assignment, the simulation generates a smaller mean error.

6 Computing

Metropolitan areas typically have 10 million or more inhabitants. Simulating all these inhabitants individually, as one does with an agent-based simulation, is a considerable computational challenge. We counter this challenge on several levels.

On the level of the traffic simulation, we use high performance parallel computing. This is done by segmenting the geographical area into many domains, and each domain is given to a different computer to simulate. The different computers communicate with each other via messages. Since these messages are exchanged many times per second, a high performance communication network is helpful, although useful results can also be produced on a cluster of standard PCs coupled by standard Ethernet Local Area Network technology.

Fig. 7 shows computational speed results for this kind of technology, for a queue simulation as explained in Sec. 2.2 and for the Switzerland morning rush-hour (6-9 AM) scenario as explained in Sec. 5. The left plot shows the real time ratio (RTR), i.e. how much faster than reality the simulation runs. An RTR of 200, for example, means that 24 hours of traffic can be computed in less than ten minutes – for all of Switzerland! The right plot shows speed-up, i.e. how much faster than a single-CPU simulation the parallel simulation is. As one can see from the figure, real time ratio and speed-up are related by a simple vertical shift in the logarithmic plot. One also notices that with 64 CPUs a speed-up of about 27 can be reached.

A separate problem was encountered with the hybrid approach (2D space, see Sec. 2.3) for the pedestrian simulation. Large scale scenarios, such as an area of, say, $50\ km \times 50\ km$, result in several millions or even billions of cells, and even several GBytes of memory are not enough to allocate memory for all those cells. We therefore implemented a lazy initialization algorithm, where the memory for the cell is only allocated on request, i.e. when for the first time a pedestrian enters the cell. Also, when a cell has not been used for a certain number of time steps, the memory is freed up again. Fig. 8 shows a demonstration scenario, and the cells which are allocated at this point in the simulation. Since typically large parts of the region are rarely or never touched by pedestrians, this approach makes it possible to use the two-dimensional cell-based approach even for large scale scenarios.

On the level of the strategic decisions (route planning, activity generation), the typical technology is to write the traffic simulation results to a file-based database and then to base the agent adaptation on this information (Raney and Nagel, 2003). Since that approach is slow when used for large scale scenarios, we are currently implementing an approach where all relevant data is permanently kept in computer memory. This implies the interaction of even more computers, some of them now being responsible for the computation of the strategic/tactical decisions. Another advantage of this approach, besides its improved speed, is that it should now be straightforward to add within-trip replanning to the parallel simulation, i.e. that agents can also make strategic decisions while they are on the road and not just in between trips (see Fig. 4 for a visual sketch of this). Preliminary results indicate that the computing time for this module for the Switzerland scenario improves from about 45 minutes to about 10 minutes. – The combined result of these two approaches is that we can indeed run metropolitan scenarios with 10 million agents, including 50 learning iterations, over night.

7 Summary

This paper explained the most important aspects of a multi-agent simulation approach to transportation and mobility planning. These aspects include a simulation of the physical system, modules which compute strategic/tactical decisions of the agents, and a learning framework which models how agents adapt to the system and to each other. It was then shown how far we are with a real-world implementation of such a system. Since such real-world systems typically are very large (several millions of agents), computing aspects were discussed, indicating that it is possible to simulate such systems in acceptable amounts of time.

As pointed out in the introduction, it is believed that such a microscopic approach to mobility simulation offers important methodological advantages over the more traditional aggregated approaches. In particular, it is possible to model each traveler's decision individually, while taking into account specific attributes such as income, or distance to public transit. The fact that this is now feasible makes it necessary to better understand the individual decision processes which generate the dynamics of the transportation system as a whole. This is an important research task for the coming years.

Acknowledgments

We would like to acknowledge the contributions of K. Axhausen, D. Cavens, N. Cetin, C. Gloor, L. Mauron, and M. Vrtic. Our computational system is excellently maintained by the "Informatics support group" of the Department of Computer Science, in particular by M. Schmitt, for which we are particularly grateful. We also thank the Swiss federal administration which provided the real-world input data.

References

Barrett, C. L., R. Jacob and M. V. Marathe (2000): Formal-language-constrained path problems. SIAM J COMPUT, Vol. 30, No. 3, 809–837.

Beckman, R. J., K. A. Baggerly and M. D. McKay (1996): Creating synthetic base-line populations. Transportion Research Part A – Policy and Practice, Vol. 30, No. 6, 415–429.

Bowman, J. L. (1998): The day activity schedule approach to travel demand analysis. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Bundesamt für Strassen (2000): Automatische Strassenverkehrszählung 1999. Bern, Switzerland.

Charypar, D. (2002): Genetic algorithms for activity planning. Term project, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. See www.inf.ethz.ch/ nagel/papers.

Doherty, S. T. and K. W. Axhausen (1998): The developement of a unified modelling framework for the household activity-travel scheduling process. In Verkehr und Mobilität, Technical University, Aachen, Germany: Institut für Stadtbauwesen, No. 66 in Stadt Region Land.

Esser, J. and K. Nagel (2001): Iterative demand generation for transportation simulations. In The Leading Edge of Travel Behavior Research, edited by D. Hensher and J. King, Oxford: Pergamon, 689–709.

Ferber, J. (1999): Multi-agent systems. An Introduction to distributed artificial intelligence. Addison-Wesley.

Gawron, C. (1998): An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. International Journal of Modern Physics C, Vol. 9, No. 3, 393–407.

Gazis, D.C. (1974): Modeling and optimal control of congested transportation systems. Networks, Vol. 4, 113–124.

Goldberg, D.E. (1989): Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

Hofbauer, J. and K. Sigmund (1998): Evolutionary games and replicator dynamics. Cambridge University Press.

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

Meyer-König, T., H. Klüpfel and M. Schreckenberg (2001): Assessment and analysis of evacuation processes on passenger ships by microscopic simulation. In Pedestrian and Evacuation Dynamics, edited by M. Schreckenberg et al, Springer, 297–302.

Raney, B. and K. Nagel (2003): Truly agent-based strategy selection for transportation simulations. Paper 03-4258, Transportation Research Board Annual Meeting, Washington, D.C. Also see sim.inf.ethz.ch/papers.

Russel, S.J. and P. Norvig (1995): Artificial intelligence: a modern approach. Series in Artificial Intelligence, Prentice Hall.

Sheffi, Y. (1985): Urban transportation networks: Equilibrium analysis with mathematical programming methods. Englewood Cliffs, NJ, USA: Prentice-Hall.

Stein, D.L. and others, editors (since 1988): Lectures in the sciences of complexity. Santa Fe Institute in the sciences of complexity, Addison-Wesley.

von Neumann, J. (1966): Design of computers, theory of automata and numerical analysis, Vol. 5 of *Collected Works of John von Neumann*. Univ. of Illinois Press.

Vrtic, M. and K.W. Axhausen (2002): Experiment mit einem dynamischen umlegungsverfahren. Strassenverkehrstechnik. Also Arbeitsberichte Verkehrs- und Raumplanung No. 138, see www.ivt.baug.ethz.ch.

Vrtic, M., R. Koblo and M. Vödisch (1999): Entwicklung bimodales Personenverkehrsmodell als Grundlage fr Bahn2000, 2. Etappe, Auftrag 1. Report to the Swiss National Railway and to the Dienst für Gesamtverkehrsfragen, Prognos AG, Basel. See www.ivt.baug.ethz.ch/vrp/ab115.pdf for a related report.

Wolfram, S. (1986): Theory and Applications of Cellular Automata. Singapore: World Scientific.

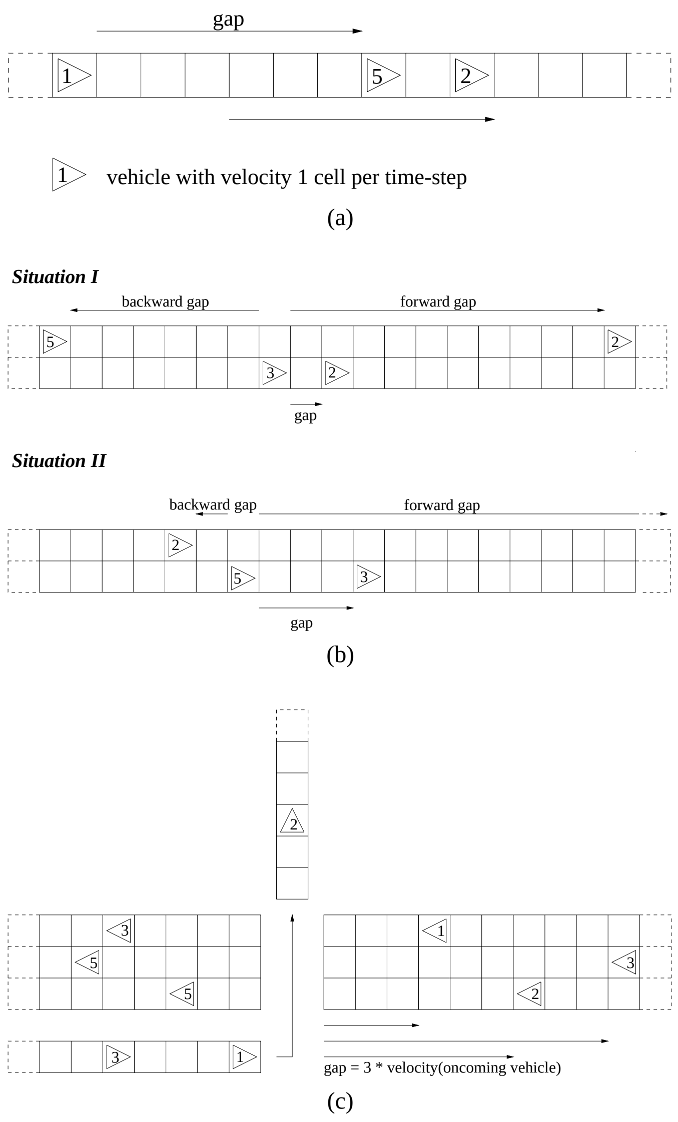
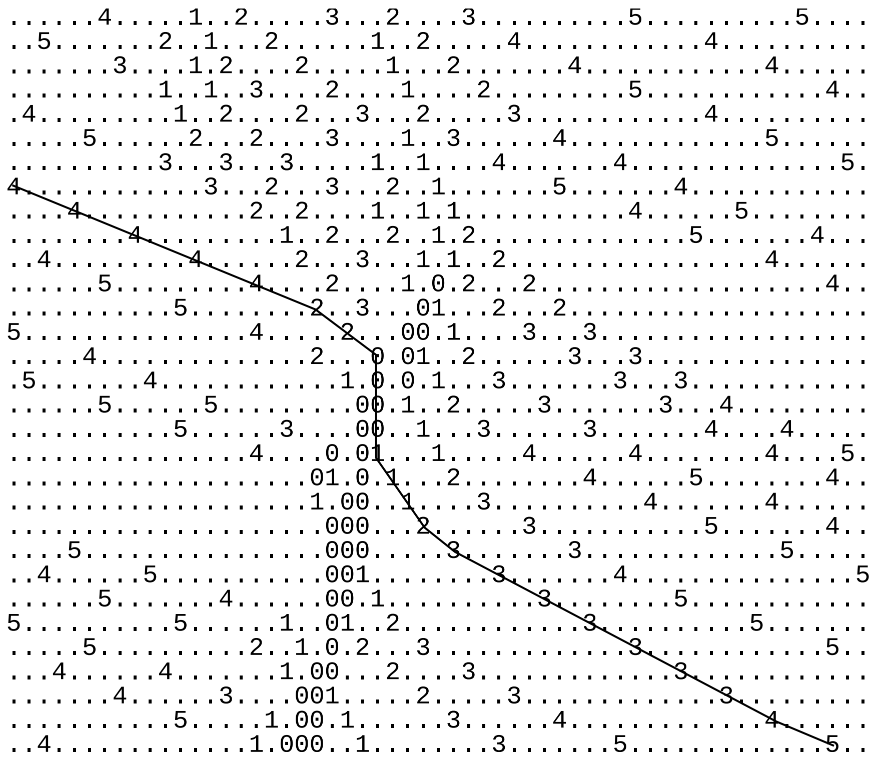


Figure 1: Illustration of cellular automata driving rules. *(a)* Definition of *gap* and one-lane update. *(b)* Lane-changing. *(c)* Left turn against oncoming traffic.



```
......4.....1..2.....3...2....3..........5..........5....
..5.......2..1...2......1..2.....4............4..........
.......3....1.2....2.....1...2.......4............4......
..........1..1..3....2....1....2.........5............4..
.4.........1..2....2...3...2.....3............4..........
.....5......2...2....3....1..3......4.............5......
..........3...3...3.....1..1....4.......4..............5.
4............3...2...3...2..1.......5.......4............
....4...........2..2....1..1.1...........4......5........
........4.........1..2...2..1.2..............5.......4...
..4.........4......2...3...1.1..2.................4......
......5.........4....2....1.0.2...2...................4..
...........5........2..3...01...2...2....................
5...............4.....2...00.1....3...3..................
.....4..............2...0.01..2......3...3...............
.5.......4............1.0.0.1...3.......3...3............
......5......5.........00.1..2.....3.......3...4.........
...........5......3....00..1...3......3.......4....4.....
................4....0.01...1.....4......4........4....5.
....................01.0.1...2........4......5........4..
....................1.00..1....3..........4.......4......
.....................000...2......3...........5.......4..
....5................000.....3.......3.............5.....
..4......5...........001........3.......4...............5
......5.......4......00.1..........3........5............
5..........5......1..01..2............3..........5.......
.....5..........2..1.0.2...3.............3............5..
...4......4.......1.00...2....3.............3............
.......4......3....001.....2.....3.............3.........
...........5.....1.00.1......3......4.............4......
..4.............1.000..1........3.......5.............5..
```

Figure 2: Stochastic CA. Jam out of nowhere leading to congested traffic.

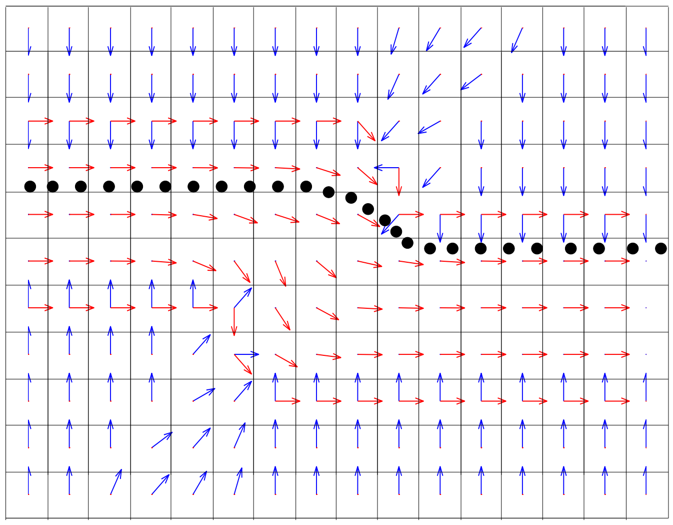


Figure 3: The hybrid simulation technique. The forces (arrows) are valid for the whole cell; a pedestrian's trajectory (dots) can follow arbitrary positions.

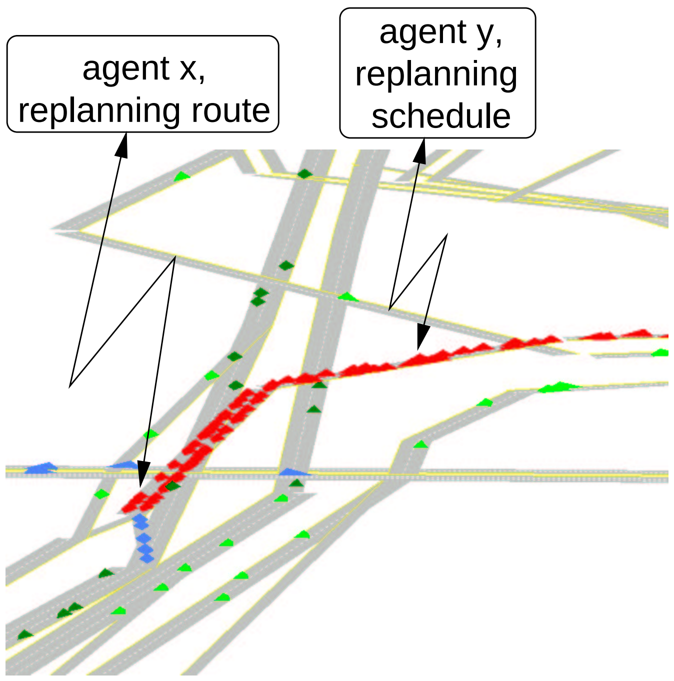


Figure 4: Virtual reality representation of simulated traffic in Portland/Oregon. Including visualization of within-day replanning.

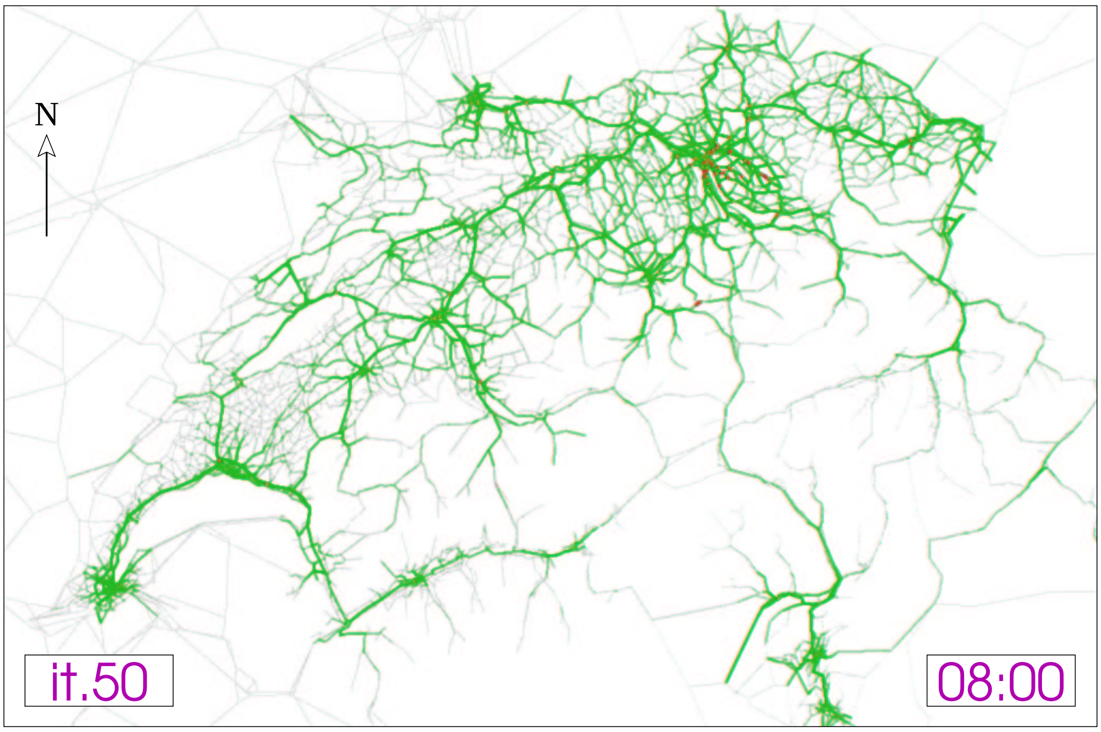
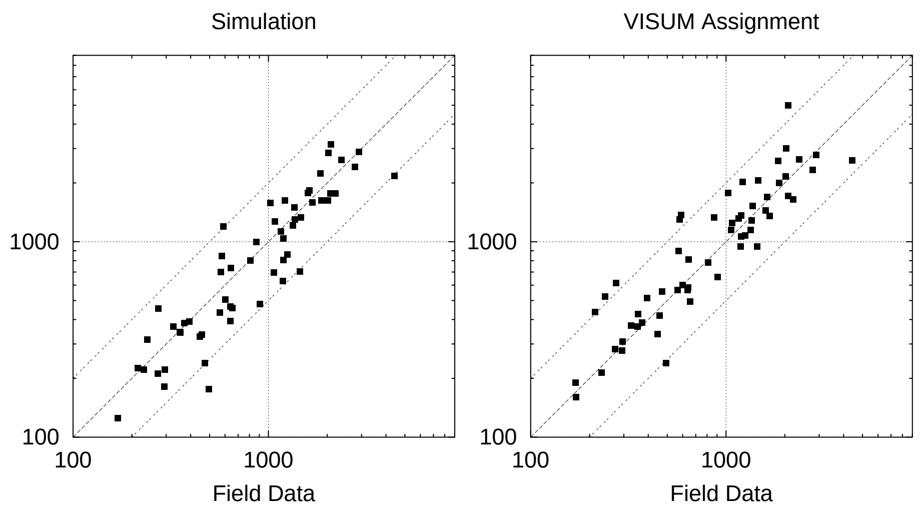


Figure 5: Snapshot of Switzerland at 8:00 AM. From the queue micro-simulation, iteration 50.



(a) (b)

Figure 6: Comparison of link volumes for 7:00-8:00 AM from *(a)* Simulation (iter. 50) or *(b)* VISUM assignment model, vs. corresponding hourly counts from field data.

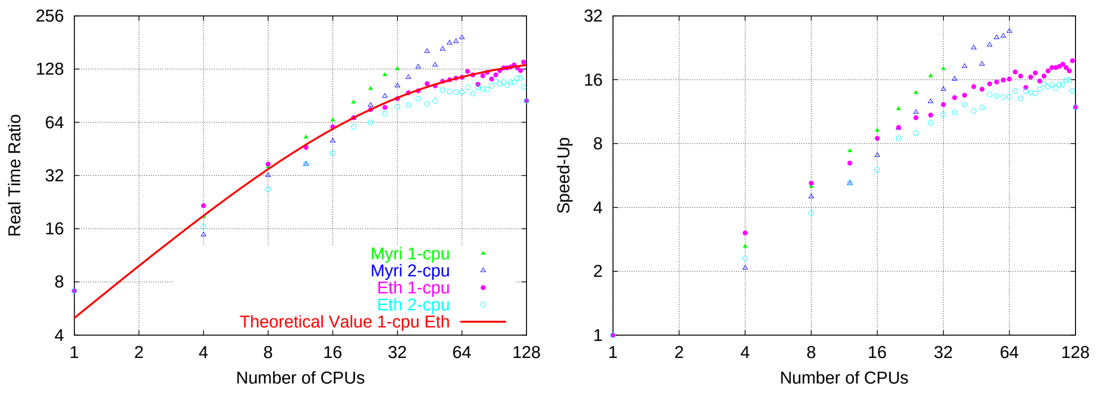


Figure 7: RTR (left) and speed-up (right) for the Switzerland morning rush-hour scenario on single and dual CPU machines, using Ethernet or Myrinet.

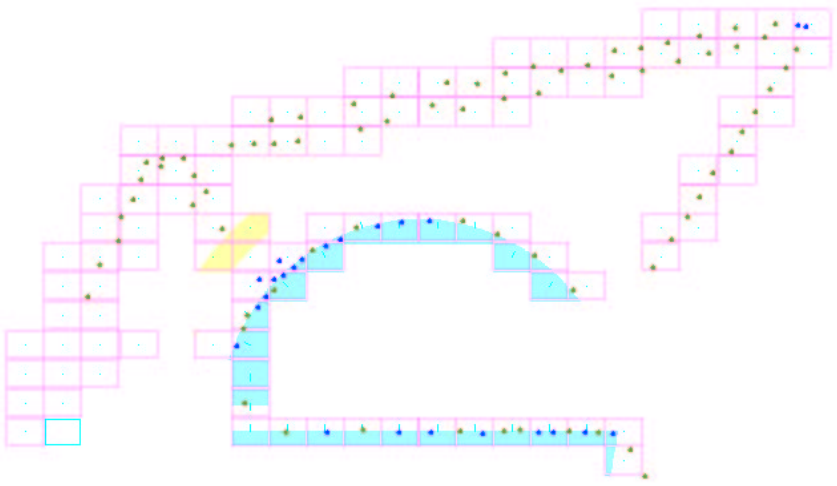


Figure 8: Dynamic memory allocation for hybrid pedestrian simulation

Table 1: Bias and Error of Simulation and VISUM Results Compared to Field Data

	Simulation	VISUM
Mean Abs. Bias:	-64.60	$+99.02$
Mean Rel. Bias:	-5.26%	$+16.26\%$
Mean Abs. Error:	263.21	308.83
Mean Rel. Error:	25.38%	30.42%