# 1 Route learning in iterated transportation simulations

Kai Nagel

Dept. of Computer Science, ETH Zürich, 8092 Zürich, Switzerland, nagel@inf.ethz.ch

**Abstract.** Transportation simulation packages need to generate the routes along which vehicles move through the network, and these routes need to be sensitive to congestion. The traditional solution to this problem is static assignment. Unfortunately, static assignment does not work when confronted with more realistic dynamics such as spatially extended and/or time-varying queues. This paper looks at issues which come up when moving away from the static, flow-based representation toward a dynamic, agent-based representation. The important difference is that learning and adaptation are moved away from the system and toward the agents. There is however rather a smooth crossover than a sharp transition between the two views, and intermediate methods are possible, with trade-offs between fast relaxation vs. realistic modeling of human behavior.

## 1.1 Introduction

Many contributions to this book are about route selection of humans in transportation networks. There are experimental contributions – where humans are observed how they solve this problem – and modeling contributions – where computer simulations attempt to generate plausible routes.

One area where such modeling knowledge is needed is the area of transportation simulations. Such simulations are built for many purposes, from signal coordination for traffic management to long-term regional planning. In particular for long-term planning, it is clear that a simulation of traffic and cars alone is not enough, and strategic behavior of the agents including activity and route choice need to be included. There is an emerging consensus that such transportation simulation packages should consist at least of the following modules:

- **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.
- 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 will probably be necessary to generate good default layouts for intersections etc. without always knowing the exact details.

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.

The modules interact, and the interaction goes in both directions: for example, 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.

There are two ways to see this relaxation mechanism: as a solution method to a nonlinear optimization problem, or as modeling human learning. In the first interpretation, the assumption is that the relaxed state, which typically is a Nash Equilibrium, is how the real system behaves. As we will see later, this interpretation normally goes along with a mathematical formulation for which one can prove uniqueness of the solution, and the computation should get there as quickly as possible. In this approach, the agents do not learn explicitly. All learning happens outside the modeling; the modeling is only interested in the final state of the system. This is fundamentally different from the second approach, which models human learning directly. Here, the computation specifies learning rules for each individual agent, and the simulation is run repeatedly to allow for day-to-day learning.

There is however no clear dividing line between both interpretations. For example, for some systems and some methods of explicit learning one can show that they converge to the same Nash Equilibrium as the solution to the nonlinear optimization problem. Conversely, some computational methods to solve the nonlinear optimization problem in fact resemble human learning. Often, a method which models human learning is used, but in the computation learning is made much faster than in reality in order to save on computer time.

In this contribution, we will focus on simulations of such learning behavior, in particular with respect to route choice, although many of the arguments should also apply to other aspects. In particular, we will look at the following:

- Relations between human learning and static assignment (Sec. 1.2)
- Basic implementations of learning via feedback; convergence; structure of the search landscape (Sec. 1.3)
- Individualization of knowledge and resulting robustness of the implementation (Sec. 1.4)
- Day-to-day vs. within-day replanning (Sec. 1.5)
- Smarter agents, system performance, and unpredictability (Sec. 1.6)

The paper is concluded by a summary.

## 1.2 Traffic assignment

The traffic assignment – or route assignment – problem can be formulated as follows: Given a set of trips via starting times, starting locations, and destinations, find a "good" route for each trip. This is in fact a strong simplification of the real world situation, where neither the starting times nor the trips themselves are fixed. It is however a useful starting point for analysis. Even for this simplified problem, there are many variations. Some of them will be described in the following.

### 1.2.1 Static assignment

This is the maybe best-known formulation of the problem (e.g. [1]). Instead of single trips, one assumes constant traffic streams $r_{ij}$ going from each origin $i$ to each destination $j$. The collection of $r_{ij}$ is also called the origin-destination (OD) matrix. Routes from $i$ to $j$ are numbered by $k$; $r_{ij,k} \geq 0$ is the number of trips using the $k$-th route. Let $\delta_{ij,k,a}$ be an indicator if route $ij,k$ uses link $a$. The number of trips using link $a$ then is

$$x_a = \sum_{ij} \sum_k r_{ij,k} \, \delta_{ij,k,a} \ .$$

The link travel time (link cost) is normally defined via a function $t_a(.)$. The trip time of a route in consequence is

$$T_{ij,k} = \sum_a t_a(.) \, \delta_{ij,k,a} \ .$$

The problem specification now is that $r_{ij,k}$ need to be found such that

$$\sum_k r_{ij,k} = r_{ij}$$

and such that all used routes have the same travel time and no unused route has a faster ($=$ better) travel time.

If one assumes that $t_a$ depends on the flow $x_a$ only (i.e. neither on other aspects of traffic flow nor on traffic on other links), and if $t_a(x_a)$ is strictly monotonic, then it can be shown that the static assignment problem has a unique solution, and in consequence a method with a fast convergence rate should be selected. Since the problem is typically non-linear, these methods are relaxation methods, that is, a partial solution is constructed, which is continuously improved through the algorithm.

### 1.2.2 Dynamic OD matrices

In static assignment, the basic assumption is that the traffic streams $r_{ij}$ are constant. In reality, however, these traffic streams are time dependent; for example, traffic streams in the morning are clearly different from traffic streams in the evening. A possibility to deal with this in the framework of static assignment is to have different $r_{ij}(t)$ for different time periods, and to solve separate static assignment problems for each time period. Each of these separate static assignment problems solves the problem as if the $r_{ij}(t)$ were constant; in consequence, such an approach is valid only when real-world traffic is sufficiently steady-state over a longer period of time.

### 1.2.3 Static spatially extended queues

If, for a bottleneck, demand is larger than capacity, the result is that a queue forms upstream of the bottleneck. Under steady state conditions such as in static assignment, the waiting time in queues is a component of the travel time. In standard static assignment, as explained above, queues are assumed to have no spatial extension ("point queues"), and as long as the network has enough overall capacity, there will be an equilibrium solution. The "enough overall capacity" constraint is automatically fulfilled with link cost functions which admit arbitrarily high link flows (albeit at a high cost), such as the BPR link cost function.

The situation changes dramatically once links have hard capacity constraint, and traffic forms spatially extended queues upstream of bottlenecks. Although steady state equilibrium solutions to such scenarios are possible, they may not be unique, and possibly one solution does fulfill the demand set by the OD matrix and another one does not [2]. That is, with spatially extended queues already the static problem leaves the area of where a good mathematical foundation is available.

### 1.2.4 Dynamic spatially extended queues

None of the problem formulations so far deals with the problem of dynamic queues, i.e. the dynamic reality that queues grow and shrink. Dynamic queues leave the framework of static assignment, and one normally resorts to simulation. In order to solve the resulting *dynamic* traffic assignment (DTA) problem, one uses a relaxation method: All travelers select a route, the simulation is run, some travelers adjust their route, the simulation is run again, etc., until some convergence criterion is fulfilled.

Interestingly, this is still similar to typical static assignment solution procedures: there is a routing procedure which gives routes to each individual traveler, and there is a simulation ("network loading") which returns the link travel times $t_a$ based on the routes. Since a simulation is used to generate those links travel times, one is however much less restricted in terms of the choice of the particular traffic dynamics than with traditional static assignment.

There is much less knowledge about the types of solutions that such an approach will generate. One way to look at such iterated systems is to treat them as time-discrete dynamical systems [3]. A state is the trajectory of the simulation through one day; an iteration is the update from one day (period) to the next (Fig. 1.1). As such, one can search for fix points (for deterministic systems), steady state densities (for stochastic systems), 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 [4], although for the computationally feasible number of iterations the possible occurrence of "broken ergodicity" [5] 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.

### 1.2.5 Human behavior and agent-based simulation

So far, the assumption was that we attempt to find a Nash equilibrium, although the situation has become more and more complex. It is however not automatically clear that

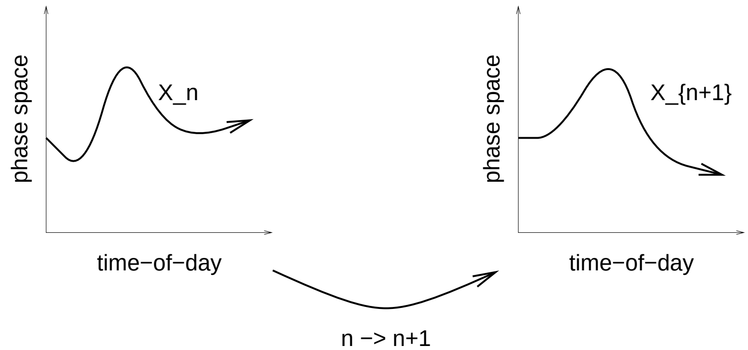


**Fig. 1.1.** Schematic representation of the mapping generated by the feedback iterations. Traffi c evolution as a function of time-of-day can be represented as a trajectory in a high dimensional phase space. Iterations can be seen as mappings of this trajectory into a new one.

reality fulfills a Nash equilibrium; rather, we have many travelers making individual decisions which may or may not lead to a Nash equilibrium. In agent-based simulations, it is possible to implement this directly: The simulations have individual travelers, and they behave according to certain rules. Now, convergence to Nash equilibrium is a possible outcome of the dynamics, not a requirement.

Also, as indicated in the introduction, route choice is in fact only a subset of all possible decisions that a traveler is faced with. When adding other choice dimensions, such as activity selection or even housing choice, the situation becomes even more complicated, and the framework of route assignment can no longer be maintained. For example, the housing market is never relaxed, and thus a behavioral modeling of the transients is a necessity [6]. Also, increasingly more complex aspects of the human decision-making means that we are not able to compute a "best" solution – and we suspect that humans in the real world do not do this, either. All this points to the use of multi-agent simulations, where agents are represented by behavioral rules, which can be changed directly. Aspects of such simulations will be discussed in the next sections.

## 1.3 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. As said before, widely accepted method to resolve this is systematic relaxation (e.g. [3]) – 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. The method is somewhat similar to a standard relaxation technique in numerical analysis.

Fig. 1.2 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 indicated 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 [7], which is a queuing 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 vehicles can enter, causing spill-back. Compared to the original version of Ref [8], our model has an improved intersection dynamics [7]. 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.
2. 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.
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.

The routing is done 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. That is, for each link $a$, there is a time-dependent cost function, $c_a(t)$, which we obtain as a piece-wise constant approximation of the averaged link travel times returned from the simulation. When a standard Dijkstra algorithm is used in the forward direction, i.e. starting at the starting point of the trip at the corresponding starting time and expanding into all directions as standard in Dijkstra, one just has to replace the fixed link costs by $c_a(t)$ in the standard algorithm. In principle, the time dependency adds the possibility that a faster route could be obtained by waiting at a node, but since this is not realistic for car traffic, this possibility can be ignored.

Fig. 1.2 right is the result after 49 such iterations. Quite visibly traffic has spread out over many more different routes.

Fig. 1.3 shows the relaxation behavior for a scenario in Dallas [9,10]. 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.[1]

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 a “best reply” dynamics (i.e. find the best answer to yesterday’s traffic), and for some (non-traffic) systems it can even be proven that this converges to a Nash equilibrium [11].

As mentioned above, traffic has a tendency to spread out in reaction to congestion. That is, people shift from congested to uncongested links, even if under uncongested conditions the new route would take more time. One would expect that one could note

[1] In fact, we have observed the multi-stabilities corresponding to the effects in Sec. 1.2.3 in our simulations. However, our stochastic simulations seem to take averages over them. Further research would be useful.

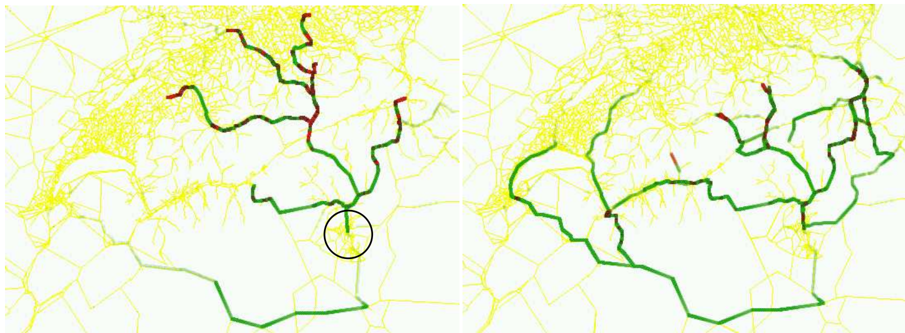

**Fig. 1.2.** Result of day-to-day learning in a test example. The scenario is a test case, using a network of Switzerland, where 50 000 vehicles from all over the system start between 6am and 7am and simultaneously attempt to reach a destination within the circle. Green shows free flowing traffi c; red shows traffi c jams. LEFT: Situation at 9:00am in the zeroth iteration. RIGHT: Situation at 9:00am in the 49th iteration. Clearly, traffi c has spread out and the system is using more different routes in the 49th iteration.

this in the strategy landscape, i.e. that the performance difference between the first $k$ fastest path is smaller under congested than under uncongested conditions. This conjecture is in fact correct [12]. Fig. 1.4 (left) shows performance decrease incurred by selecting the $k$-best route instead of the best one, once for an uncongested network and once for a congested one. Clearly, the strategy landscape is much flatter in the congested situation. It is however equally flat in the relaxed and in the unrelaxed situation. Fig. 1.4 (right) shows that the additional options under relaxed congested conditions are truly different from the best option, which is not true for the unrelaxed congested situation. This, in summary: Under uncongested conditions, deviating from the fastest path incurs a high penalty. Under unrelaxed congested conditions, there are many good options, but they are all similar to each other. Only under relaxed congested conditions do we find many truly different options with similar performance.

The above results were obtained with a Dallas/Fort Worth scenario within the TRANSIMS project. "Uncongested" refers to the empty network, where free speeds were used to compute link travel times. "Unrelaxed congested" refers to the zeroth iteration traffic in the middle of the rush hour. "Relaxed congested" refers to a situation after many iterations; in fact, link travel times were averaged over 10 iterations. All curves in Fig. 1.4 refer to an average over 955 randomly selected OD (origin-destination) pairs in the study area which are between 7 km and 7.01 km apart.

## 1.4 Individualization of knowledge

Knowledge of agents should be private, i.e. each agent should have a different set of knowledge items. For example, real people only know a relatively small subset of the street network ("mental map"), and they have different knowledge and perception of congestion (e.g. [14]).

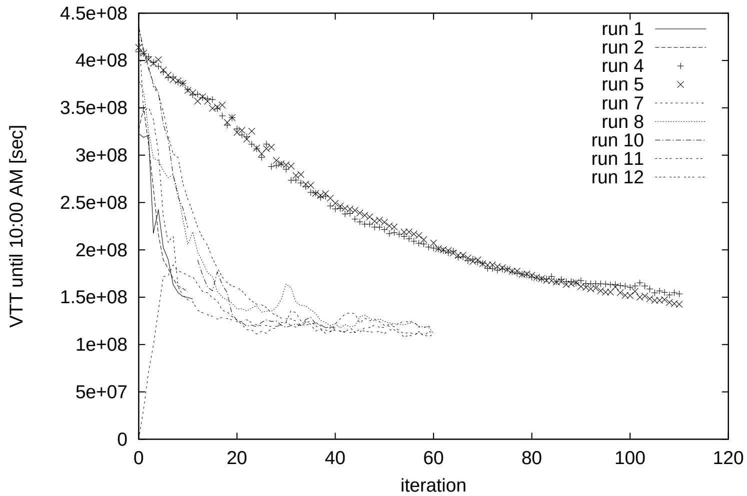


**Fig. 1.3.** Different relaxation paths in day-to-day replanning. The plot shows the sum of all travel times as a function of the iteration for different relaxation methods. All methods seem to lead to the same relaxed state. From [13].

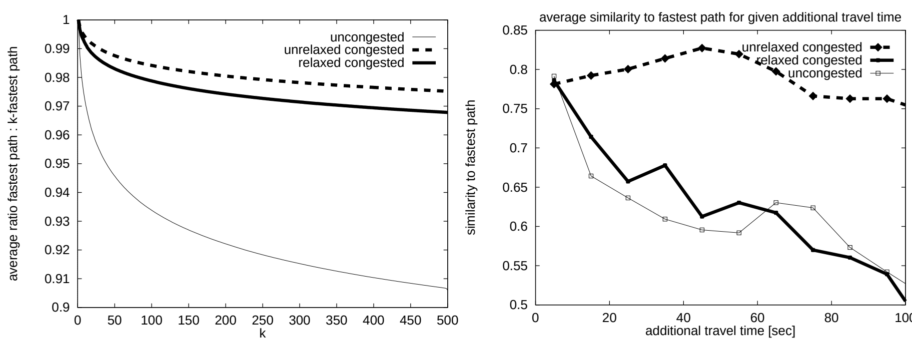


**Fig. 1.4.** Flatness of the strategic landscape for route choice behavior. LEFT: Ratio of the travel times of the fastest path to the $k$-fastest path as a function of $k$. Clearly, this measure decreases more quickly in an uncongested than in a relaxed congested network, meaning that in a relaxed congested network, there are many more alternatives at about the same system performance. RIGHT: Under relaxed congested conditions, the additional options are truly dissimilar. Shows is the similarity to the best option as function of the additional travel time. – From [12].

This opens the door for the use of Complex Adaptive Systems methods (e.g. [15]). 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 [16]):

- Plans evaluation. In this phase, plans (or strategies) need to be 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 [17,18].
- Plans generation. In this phase, new plans (or strategies) need to be generated.

A major advantage of this approach is that it becomes more robust against artifacts of the router: if an implausible route is generated, the agent will fall back on a more plausible route generated earlier. Fig. 1.5 shows an example. The scenario is the same as in Fig. 1.2; 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 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 will 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 [16]. Bad routes will be weeded out via the performance evaluation method. For more details see [19]. Other implementations of partial aspects are [20–22,14].

The way we have explained it, one will probably assume that each individual has 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 our Switzerland simulations with a network of $28\,622$ links, we have $N_{people} \sim 7.5$ mio, $N_{trips} \sim 3$, $N_{links} \sim 50$, and $N_{options} \sim 5$, which results in

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

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

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 each possible destination. Each node in the network has "signposts" for which way to go for any possible destination; 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 different 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

$$1\,000 \times 10\,000 \times 5 \times 4 = 200 \text{ MByte} ,$$

considerable smaller 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 the 28 622 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. The memory requirements for the network-oriented plans storage now become

$$28\,622 \times 10\,000 \times 100 \times 5 \times 4 \approx 600 \text{ GByte} ,$$

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 the information by a factor of at least 30 [23].

## 1.5 Within-day re-planning

Day-to-day replanning still assumes, in some sense, "dumb" particles. Particles follow routes, but the routes are pre-computed, and once the simulation is started, they cannot be changed, for example being adapted 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 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 through time. And indeed, many dynamic traffic assignment (DTA) systems work exactly in this way (e.g. [3]), in spite of several problems in particular with quick congestion build-up [19].

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 example, 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

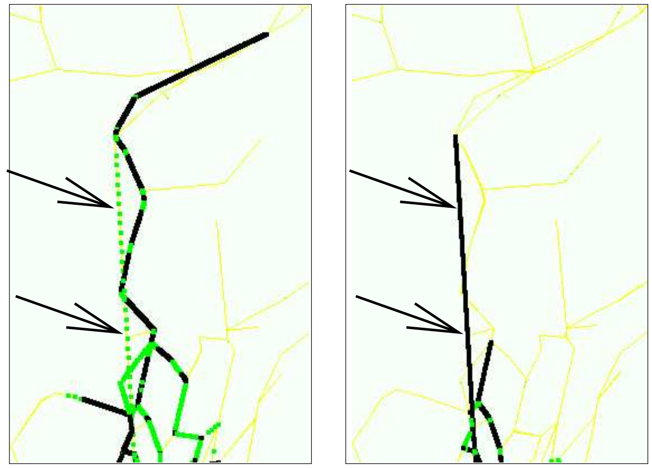

**Fig. 1.5.** Individualization of plans and interaction with router artifacts. LEFT: All vehicles are re-planned according to the same information; vehicles do not use the freeway (arrrows) although the freeway is empty. As explained in the text, this happens because the router makes erroneous predictions 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. – The arrows point to the freeway. The time is 7pm.

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 [24,13] and for the realistic modeling of certain aspects of human behavior [25,26] has been pointed out. Nevertheless, we are not aware of operational implementations of this aspect.

## 1.6 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. 1.6, 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

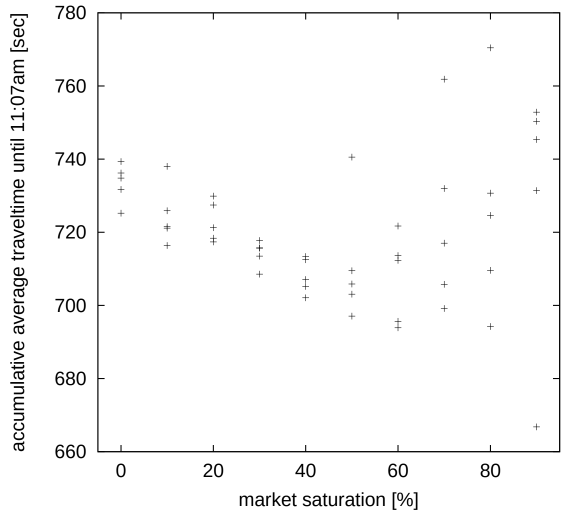


**Fig. 1.6.** Predictability as function of within-day rerouting capabilities. The result was obtained in the context of a simulation study of route guidance systems. The x-axis shows the fraction of equipped vehicles; the y-axis shows average travel time of all vehicles in the simulation. For each value of market saturation, fi ve 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 [13].

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 [27]. 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. [28] 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.

## 1.7 Summary

Simulations need knowledge about human behavior in order to feed the simulation models with realistic rules. In that sense, simulation projects are customers of the results of most of the research presented in this book. Simulations are however not simple translations of human behavior into computer models – rather, modeling, being “as much an art as a science”, often involves reducing the evidence provided by psychological experiments into a few simple rules. This is particularly true for large scale simulations, where the computation of individual strategic decisions cannot take more than a few seconds per agent. The restriction of computational resources also explains why large scale simulations put as much emphasis on fast relaxation than on realistic modeling on human behavior. This emphasis is reinforced by the fact that fast relaxation of agent-based simulations is similar to fast relaxation of traditional assignment.

This paper then looks at some issues of true agent-based implementations, such as agent-oriented plans storage, or an agent database to remember more than one strategy per agent. These modifications move iterated transportation simulations away from the framework of non-linear optimization and into the realm of complex adaptive systems. As pointed out in the text, these changes sometimes make the simulations more robust and more stable, as does the agent database. Often however, they make the simulations less stable in the sense of larger fluctuations of Monte Carlo runs. This seems to be particularly true when agents become “smarter”, meaning that they in some sense use more knowledge about the system.

## References


1. Y. Sheffi. *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1985.
2. C.F. Daganzo. Queue spillovers in transportation networks with a route choice. *Transportation Science*, 32(1):3–11, 1998.
3. J.A. Bottom. *Consistent anticipatory route guidance*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.
4. E. Cascetta and C. Cantarella. A day-to-day and within day dynamic stochastic assignment model. *Transportation Research A*, 25A(5):277–291, 1991.
5. 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.
6. P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson. Microsimulation of urban development and location choices: Design and implementation of UrbanSim. *Networks and Spatial Economics*, in press.
7. N. Cetin and K. Nagel. Parallel queue model approach to traffic microsimulations. In *Swiss Transport Research Conference*, Monte Verita, Switzerland, March 2002.
8. 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.
9. 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.
10. 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.
11. J. Hofbauer and K. Sigmund. *Evolutionary games and replicator dynamics*. Cambridge University Press, 1998.
12. T. Kelly and K. Nagel. Relaxation criteria for iterated traffic simulations. *International Journal of Modern Physics C*, 9(1):113–132, 1998.

13. M. Rickert. *Traffic simulation on distributed memory computers*. PhD thesis, University of Cologne, Germany, 1998. See www.zpr.uni-koeln.de/˜mr/dissertation.
14. S. Weinmann. *Simulation of spatial learning mechanisms*. PhD thesis, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, in preparation.
15. J. H. Holland. Using classifi er systems to study adaptive nonlinear networks.
16. M. Ben-Akiva. Route choice models. Presented at the Workshop on "Human Behaviour and Traffi c Networks", Bonn, December 2001.
17. K. Nagel and C.L. Barrett. Using microsimulation feedback for trip adaptation for realistic traffi c in Dallas. *International Journal of Modern Physics C*, 8(3):505–526, 1997.
18. K. Nagel. *High-speed microsimulations of traffic flow*. PhD thesis, University of Cologne, 1994/95.
19. B. Raney and K. Nagel. Issues of feedback routing in iterated transportation simulations. in preparation.
20. 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.
21. H. Unger. *Modellierung des Verhaltens autonomer Verkehrsteilnehmer in einer variablen staedtischen Umgebung*. PhD thesis, TU Berlin, 2002.
22. Chr. Gloor. Modelling of autonomous agents in a realistic road network (in german). Diplomarbeit, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, 2001.
23. B. W. Bush. Personal communication.
24. J. Esser. *Simulation von Stadtverkehr auf der Basis zellularer Automaten*. PhD thesis, University of Duisburg, Germany, 1998.
25. 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.
26. S. T. Doherty and K. W. Axhausen. The developement of a unifi ed 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.
27. T. Kelly. Driver strategy and traffi c system performance. *Physica A*, 235:407, 1997.
28. K. Nagel and S. Rasmussen. Traffi c 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.