# 5

# ADJUSTMENTS OF ACTIVITY TIMING AND DURATION IN AN AGENT-BASED TRAFFIC FLOW SIMULATION

*Michael Balmer, ETH Zurich, Zurich, Switzerland*

*Bryan Raney, ETH Zurich, Zurich, Switzerland*

*Kai Nagel, TU Berlin, Berlin, Germany*

## INTRODUCTION

One possibility to bring activity-based demand generation into the transportation planning processes is to use it to replace the first two (or three) steps of the conventional four step process. Activity-based demand generation would produce a standard origin-destination (OD) matrix, which would then be fed into the existing assignment model. Both the OD matrix and the assignment model could be time-dependent. The advantage of this approach is that it ties in with the arguably most sophisticated and best understood part of the four step process: route assignment. Yet, some of these advantages disappear when the OD matrices are time-dependent. In that situation, very few of the mathematical results of static assignment carry over. In addition, coupling activity-based demand generation to network assignment through an OD matrix disrupts the connection between individuals and their performance in the simulated traffic system. Any iterative feedback from the traffic system performance to the activity generation can only be based on aggregate measures, such as link travel times, not on individual performance of the traveller. An obvious case where the coupling through the OD matrix goes wrong is when it is possible for a person to complete an activity even before he/she has arrived at the destination where the activity will be conducted.

In: Progress in activity-based analysis, H.J.P. Timmermans (ed.), in press.

To avoid such situations, we propose to use a truly agent-based representation of the traffic system and the assignment process. In a truly agent-based representation, each person remains individually identifiable throughout the whole simulation process. In particular, the traffic micro-simulation assumes the role of a realistic representation of the physical system, including explicit modelling of persons walking to the bus stop, or of a bus being stuck in traffic. Also, in terms of analysis such a system offers enormous advantages. It is, for example, possible to obtain the demographic characteristics of all drivers being stuck in a particular traffic jam. It is also possible to make each traveller react individually to exactly the conditions that this traveller has experienced, rather than to aggregated conditions.

This multi-agent concept consists of basically two parts: (i) the simulation of the "physical" properties of the system, and (ii) the generation of the agents' strategies. The simulation of the physical system is the place where the agents interact with each other—car drivers produce congestion, traffic lights change their intervals dependent on the amount of traffic, pedestrians wait for the next train to catch, and so on. The agents make their strategies based on what they experienced in the physical simulation—car drivers try other routes to avoid congestion, pedestrians need to leave earlier to catch the train, traffic lights favour the main streets to maximize the throughput of an intersection, etc.

We are in the process of implementing such a multi-agent simulation for the whole of Switzerland. This paper concentrates on the Zurich area, with about 260,000 agents that cross this region. The challenges with such an implementation are many: availability and quality of input data, computational implementation and computational performance, conceptual understanding of agent learning, and validation. In previous research (Raney *et al.*, 2003), we have reported the first results based on typical transportation planning data: standard origin-destination matrices; the transportation planning network from the corresponding Swiss federal planning authority; and performed route assignment (dynamic traffic assignment or DTA) based on these input data. The two main differences with other DTA systems, such as DYNASMART (www.mit.edu/its) or DYNAMIT (www.dynasmart.com), were that our system uses individual route plans for each agent while standard DTA systems store the routing decisions in the network, and that our system was run on really large scale scenarios with several millions of travellers. A newer version of DYNASMART, however, now also uses individual routes, and other systems also move towards increasingly large scales. In contrast to TRANSIMS (www.transims.net), which has used individual routes and large scales for many years now, we used a so-called agent database, which keeps track of several plans for each agent.

This chapter goes further by now also internalizing the time structure of the input data. In other words, it is possible for the simulation system to predict when agents start and end their main

activities. The main result is that it is possible to completely ignore the time structure of the time-dependent OD matrices without compromising predictive power. This is similar to the approach used and results obtained with METROPOLIS (De Palma and Marchal, 2002). The main difference is that our implementation uses complete daily activity chains, whereas METROPOLIS only uses trips. We believe that our method, while computationally more demanding, opens the door to more flexible transportation planning models.

This chapter will continue with an outline of the general simulation structure, where we also describe in more detail the modules we will use. Next, we introduce the network and the scenario. The results of the different setups of the scenario are compared with traffic count data. Some computational issues are discussed next. The paper is concluded by a section on future work.

## SIMULATION STRUCTURE

### Overview

As pointed out before, our simulation is constructed around the notion of agents that make independent decisions about their actions. Each traveller of the real system is modelled as an individual agent in our simulation. The overall approach consists of three important pieces:

1. Each agent independently generates a so-called *plan*, which encodes its intentions during a certain time period, typically a day. As this is an application to traffic forecasting, a plan contains the itinerary of activities the agent wishes to perform during the day, plus the trips the agent must take to travel between activities. An agent's plan details the order, type, location, duration and other time constraints of each activity, and the mode, route and expected departure and travel times of each leg.

2. All agents' plans are simultaneously executed in the simulation of the physical system. In this chapter, this is a *traffic flow simulation*. In other publications, we use the term *mobility simulation* in order to emphasize that the simulation of the physical system can go beyond traffic.

3. There is a mechanism that allows agents to *learn*. In our implementation, the system iterates between plan generation and traffic flow simulation. The system remembers several plans for each agent, and scores the performance of each plan. Agents normally choose the plan with the highest score, sometimes re-evaluate plans with bad scores, and sometimes obtain new plans. Further details will be given below.

This chapter concentrates on “home” and “work” as the only activities, and “car” as the only mode. We do not distinguish between a trip (between two activities) and a leg (a part of a trip which uses exactly one mode), since the “mode change” can also be defined as an activity (which has a specified location, i.e. a train station). Each of the details described in the plan, such as activity duration, is a decision that must be made by the agent. These decisions are mutually dependent, but the decisions made by one agent are independent of those made by another. We divide the task of generating a plan into sets of closely related decisions, and each set is assigned to a separate *module*. An agent strings together calls to various modules in order to build up a complete plan. To support this “stringing”, the input to a given module is a (possibly incomplete) plan, and the output is plan with some of the decisions updated. Some possible modules are:

*Activity Pattern Generator*: Decides which activities an agent actually wishes to perform during the day, and in what order. At present, this module is not used, but we have a fixed “home-work-home” pattern for all agents.

*Activity Location Generator*: Determines where the agent will perform a particular activity. At present, this module is not used, but we have a fixed location for each agent's “home” and “work” activity.

*Activity Time Allocator*: Determines the timing attributes the agent will utilize for each activity in a plan. Activities have two possible timing attributes: “activity duration” and “activity end time”. After starting an activity, an agent performs the activity either for the length of “duration”, or until the “activity end time”, whichever comes first. Activities cannot overlap in time.

*Router*: Determines which route and which mode the agent chooses for each trip leg that connects activities at different locations.

A special feature of our approach is that users can choose any number and type of these modules as long as they generate some information that contributes to a plan. For that reason, it is easy to combine for example activity and mode choice into a single module or to add residential or workplace choice. This application will employ two modules only: “activity time allocator” and “router”. Other modules will be the topic of future work.

Once the agent's plan has been constructed, it can be fed into the traffic flow simulation module. This module executes all agents' plans simultaneously on the network, allowing agents to interact with one another, and provides output describing what happened to the agents during the execution of their plans. The modules produce dependencies. The outcome of the traffic flow simulation module (e.g., congestion) depends on the planning decisions made by the decision-making modules.

However, those modules can base their decisions on the output of the traffic flow simulation (e.g., knowledge of congestion). This creates an interdependency ("chicken and egg") problem between the decision-making modules and the traffic flow simulation module. We need these modules to be consistent with one another, and therefore we introduced feedback into the traffic flow simulation structure (Kaufman *et al.*, 1991; Nagel, 1995; Bottom, 2000). This involves an iteration cycle which runs the traffic flow simulation with specific plans for the agents, then uses the time allocator and the router to update the plans, and these changed plans are again fed into the traffic flow simulation, etc., until consistency between modules is reached.

The feedback cycle is controlled by the agent database, which also keeps track of multiple plans generated by each agent, allowing agents to reuse those plans at will. The repetition of the iteration cycle coupled with the agent database enables the agents to learn how to improve their plans over many iterations. In the following sections we describe the modules in more detail.

**Activity Time Allocator**

This module is called to change the timing of an agent's plan. At this point, a very simple approach is used which just applies a random mutation to the duration and end time of an agent's activities. More precisely, for the first activity, the activity end time is the only attribute that is specified and thus mutated, while for all other activities, the duration is what is specified and mutated. For each such attribute of each activity in an agent's plan, this module picks a random time from the uniform distribution [-30 min, +30 min] and adds it to the attribute. Any negative duration is reset to zero; any activity end time before 00:00 a.m. is reset to 00:00 a.m.. The entire plan is returned to the agent, with only the time attributes modified.

Although this approach is not very sophisticated, it is sufficient to obtain useful results. This is consistent with our overall assumption that, to a certain extent, simple modules can be used in conjunction with a large number of learning iterations (e.g., Nagel *et al*., 2004). Since each module is implemented as a "plug-in", this module can be replaced by an enhanced implementation if desired.

**Router**

The router is implemented as a time-dependent Dijkstra algorithm. It first calculates link travel times from the events output of the previous traffic flow simulation. The link travel times are aggregated into 15 minute time bins, and then used as the weights of the links in the network graph.

Apart from relatively small but essential technical details, the implementation of such an algorithm is straightforward (Jacob *et al.*, 1999). With the knowledge about activity chains, it computes the fastest path from each activity to the next one in the sequence as a function in time. It returns the entire plan, completed with updated paths, to be used by the agents for the next run of the traffic flow simulation.

## TRAFFIC FLOW SIMULATION

The traffic flow simulation simulates the physical world. It is implemented as a queue simulation (Gawron, 1998; Cetin and Nagel, 2003), which means that each street (link) is represented as a FIFO (first-in first-out) queue with three restrictions. First, each agent has to remain for a certain time on the link, corresponding to the free speed travel time. Second, a link storage capacity is defined which limits the number of agents on the link. If this capacity has been reached, no more agents can enter this link. Third, there is a flow capacity, which limits the number of vehicles that can leave the link in any given time step.

Even though this structure is indeed very simple, it produces traffic as expected and it can run directly using the data typically available for transportation planning purposes. On the other hand, there are some limitations compared to reality, i.e., the number of lanes, weaving lanes, turn connectivities across intersections or signal schedules cannot be included into this model. The output that the traffic flow simulation produces is a list of events for each agent, such as entering/leaving link, left/arrived at activity, and so on. Data for an event includes which agent experienced it, what happened, at what time it happened, and where (link/node) the event occurred. With this data it is easy to produce different kinds of information and indicators such as link travel time, trip travel time, trip length, percentage of congestion, and so on.

## AGENT DATABASE — FEEDBACK

As mentioned above, the feedback mechanism is important for making the modules consistent with one another, and for enabling agents to learn how to improve their plans. In order to achieve this improvement, agents need to be able to try out different plans and to tell when one plan is "better" than another. The iteration cycle of the feedback mechanism allows agents to try out multiple plans. To compare plans, the agents assign each plan a "score" based on how it performed in the traffic flow simulation. Essentially, each agent is running its own classifier system (e.g. Holland, 1992; Palmer *et al*., 1994). It is very important to note that our framework always uses actual plan performance for the score. This is in contrast to all other similar approaches that we are aware of

which typically feedback some aggregated quantity such as link travel times and reconstruct performance based on those (e.g., URBANSIM—www.urbansim.org; Ettema *et al.*, 2004). Because of unavoidable aggregation errors, such an approach can fail rather badly in the sense that the performance information derived from the aggregated information may be rather different from the performance that the agent in fact experienced (Raney and Nagel, 2003). The procedure of the feedback and learning mechanism is as follows:

*Initial Conditions*: Start with a plan file that specifies one complete plan for each agent. The agent database loads these plan files into the memory of the agents. Each agent marks its initial plan as the "selected" plan.

*Simulate*: The agent database sends the set of "selected" plans (one for each agent) to the traffic flow simulation. The simulation executes the plans simultaneously and outputs events.

*Process Events*: The agent database reads the events that are output by the traffic flow simulation and sends each one to the agent identified within it. Each agent uses its events to calculate the score of its "selected" plan—the one it most recently sent to the traffic flow simulation.

*Plan Pruning*: The number of plans kept in an agent's memory for reuse can be limited to $N$ plans to conserve memory. If $N$ is defined, each agent that has $P > N$ plans deletes its lowest-scoring $P - N$ plans in this step. Note that when an agent that has $N$ plans generates a new one, it temporarily keeps $N + 1$ plans until the new plan has been scored. Then, in this step, it deletes the worst plan (even if it is the newest one).

*Select Plans*: Each agent decides which plan to select for execution by the next traffic flow simulation. It chooses from the following selection options, according to the indicated probabilities:

- *(10 %) New Plan, Routes Only*: The agent sends an existing plan (chosen with equal probability among all plans in memory) to the router. The router calculates new routes in that plan based on the link travel times calculated from the events data from the most recent traffic flow simulation, and returns the updated plan. The new plan is added to the agent's memory and marked as "selected".

- *(10 %) New Plan, Times and Routes*: The agent sends an existing plan (chosen with equal probability among all plans in memory) to the activity time allocation module. This module "mutates" the durations and/or end times of all activities in the plan and returns the updated plan. The returned plan is also sent to the router for route re-planning. When it comes back

from the route re-planner, it is added to the agent's memory and marked as "selected". (Note that now 20 % of agents will have new routes, while only 10 % will have new times).

- *(10 %) Random Selection*: The agent picks an existing plan, chosen with equal probability among all plans in memory, without regard to their scores. This plan is marked as "selected".

- *(Rest) Probabilistic Selection*: The agent picks an existing plan from memory, choosing according to probabilities based on the scores of the plans. The probabilities are of the form

$$p \propto e^{bS_j} \tag{1}$$

where, $S_j$ is the score of plan $j$, and $\beta$ is an empirical constant. This is equal to a logit model from discrete choice theory. The chosen plan is marked as "selected".

The cycle returns to step 2 (simulate), and continues until the system has reached a relaxed state. At this point, there is no quantitative measure of when the system is "relaxed"; we just allow the cycle to continue until the outcome seems stable. Note that when an agent reuses an existing plan, its previous score is not forgotten, but averaged with its new score:

$$S = (1-\alpha)S_{old} + \alpha S_{new} \tag{2}$$

with the blending factor $\alpha$. This allows the agent to base plan selection on the plans' history and not only on the last iteration. With $\alpha = 0$ no score will be updated and the agents will not learn. With $\alpha = 1$ the history of a plan is neglected. Score averaging requires all plans to have an $S_{old}$, so when a new plan is generated, it is optimistically given a preliminary score equal to the score of the agent's best plan. More sophisticated approaches to agent learning are discussed in Timmermans *et al*. (2003).

## SCORES FOR PLANS

In order to compare plans, it is necessary to assign a quantitative score to the performance of each plan. In principle, arbitrary scoring schemes can be used (e.g., prospect theory by Avineri and Prashker, 2003). We used a simple utility-based approach, which is related to the Vickrey bottleneck model (Arnott *et al.*, 1993), but needs to be modified to be consistent with our approach

based on complete daily plans (Charypar and Nagel, 2003; Raney and Nagel, in press). The total score of a plan is computed as the sum of individual contributions:

$$U_{total} = \sum_{i=1}^{n} U_{perf,i} + \sum_{i=1}^{n} U_{late,i} + \sum_{i=1}^{n} U_{travel,i} \tag{3}$$

where $U_{total}$ is the total utility for a given plan; $n$ is the number of activities/trips; $U_{perf,i}$ is the (positive) utility earned for performing activity $i$; $U_{late,i}$ is the (negative) utility earned for arriving late at activity *i*; and $U_{travel,i}$ is the (negative) utility earned for travelling during trip $i$. In order to work in plausible real-world units, utilities are measured in Euro.

A logarithmic form is used for the positive utility earned by performing an activity (e.g., Axhausen, 1990b):

$$U_{perf,i}\left(t_{perf,i}\right) = \beta_{perf} \cdot t_i^* \cdot \ln\left(\frac{t_{perf,i}}{t_{0,i}}\right) \tag{4}$$

where, $t_{perf,i}$ is the actual performed duration of the activity, $t_i^*$ is the "typical" duration of an activity, and $\beta_{perf}$ is the marginal utility of an activity at its typical duration. $\beta_{perf}$ is the same for all activities, since in equilibrium all activities at their typical duration need to have the same marginal utility. $t_{0,i}$ is a scaling parameter that is related both to the minimum duration and to the importance of an activity.

If the actual duration falls below $t_{0,i}$, then the utility contribution of the activity becomes negative, implying that the agent should completely drop that activity. A $t_{0,i}$ only slightly less than $t_i^*$ means that the utility of activity $i$ rapidly decreases with decreasing $t_{perf,i}$, implying that the agent should rather cut short other activities where the utility does not decrease as quickly when reducing their duration. In this application, we use

$$t_{0,i} = t_i^* \cdot e^{-\varsigma/\left(p \cdot t_i^*\right)} \tag{5}$$

where $\varsigma$ is a scaling constant set to 10 hours, and $p$ is a priority indicator, here set uniformly to one. Note that with this specific form, $U_{perf,i}\left(t_i^*\right) = \beta_{perf} \cdot \varsigma$, independent of the activity type. This "consequence" is actually the motivation for the specific mathematical form of the activity

performance utility contribution, which was used because no better argument was available (Charypar and Nagel, in press); future research should lead to better versions.

The (dis)utility of being late is defined as:

$$U_{late,i} = \beta_{late} \cdot t_{late,i} \tag{6}$$

where, $\beta_{late} \leq 0$ is the marginal utility (in Euro/h) for being late, and $t_{late,i}$ is the number of hours late for activity $i$. To be able to calculate the utility of being late, a starting time window for the activities has to be given. The (dis)utility of travelling is defined as:

$$U_{travel,i} = \beta_{travel} \cdot t_{travel,i} \tag{7}$$

where $\beta_{travel} \leq 0$ is the marginal utility (in Euro/h) for travel, and $t_{travel,i}$ is the number of hours spent travelling during trip *i.*

At this point, our traffic flow simulation does not differentiate between "being at an activity location" (which potentially includes waiting) and "performing an activity". In consequence, the simulation makes the agent stay at the activity location for the length of "duration", no matter whether the agent can perform the activity or not. For example, when work starts at 8 a.m. but the agent arrives at 7 a.m. with a duration of 8 hours, then the agent will depart from the activity location at 7 a.m. plus 8 hours = 3 p.m.. The utility function, however, differentiates between "arrival time" and "activity start time". The "work" activity has a particular starting time (see sec. 0 for the particular value), and arriving before this time causes the agent to wait until then before actually starting the activity. This means that arriving early to an activity does not gain an agent any activity performance utility.

## VERIFICATION OF IMPLEMENTATION

We have verified that the simulation structure as described above works as we intended by running it on a simple test scenario consisting of a circular network with 2,000 agents going back and forth between home and work. All agents have the same "home" location on one side of the circle and the same "work" location on the other side. Nine routes are available between home and work, and one route is available between work and home. We ran three setups with various combinations of decision-making modules enabled:

*New plan, routes only*: The agents are only allowed to use the router module. They may do so with a 10 % probability.

*New plan, times only*: The agents are only allowed to use the activity time allocation module. They may do so with a 10 % probability.

*New plan, times and routes*: Agents may use the router module with 10 % probability, or both modules, with 10 % probability. This is just as described in the step 5 of Sec. 0 above.

The results from these three scenarios were as expected (Raney and Nagel, in press).

## INPUT DATA AND SCENARIO

### Network

The street network that is used was originally developed for the Swiss regional planning authority (Bundesamt fuer Raumentwicklung), and covered Switzerland. It was extended with the major European transit corridors for a railway-related study (Vrtic *et al.*, 1999). Some further modifications, in particular a capacity increase inside the Zurich city area, are described in Raney *et al.* (2003). The resulting network has the fairly typical size of 10,564 nodes and 28,624 links (Figure 1). Also fairly typical, the major attributes on these links are type, length, speed, and capacity.

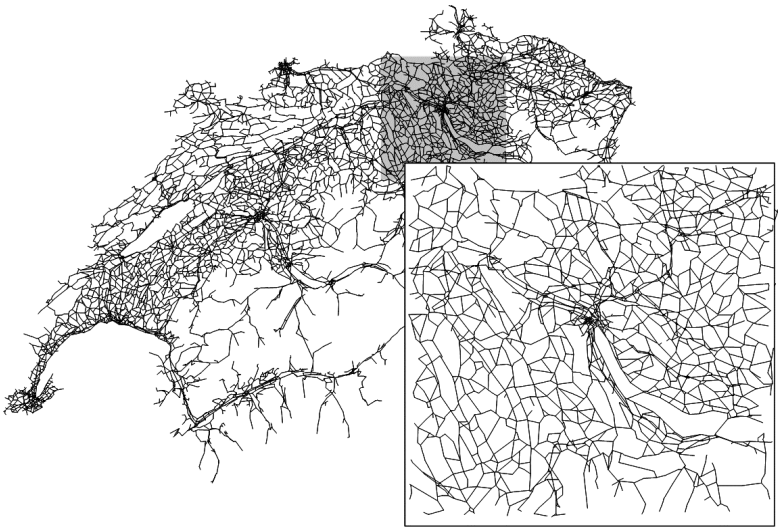


**Figure 1**
**Switzerland Network**

## ZURICH AREA SCENARIO

The full Switzerland scenario demand generation is based on 24-hour origin-destination matrices from the Swiss regional planning authority (Bundesamt fuer Raumentwicklung). The original 24-hour matrix was converted into 24 one-hour matrices using a three step heuristic (Vrtic and Axhausen, 2002). The first step employed departure time probabilities by population size of origin zone, population size of destination zone and network distance. These were calculated using the 1994 Swiss National Travel Survey (BfS, 1996). The resulting 24 initial matrices were then corrected (calibrated) against available hourly counts using the OD-matrix estimation module of VISUM (www.ptv.de). Hourly traffic count data are available from the counting stations on the national motorway system. Finally, the hourly matrices were rescaled so that the totals over 24 hours match the original 24h matrix. VISUM assignment of the matrices showed that the patterns of congestion over time are realistic and consistent with the known patterns.

For the multi-agent simulation, these hourly matrices were then disaggregated into individual trips. That is, we generated individual trips such that summing up the trips would again result in the given OD matrix. The starting time for each trip was randomly selected between the starting and the ending time of the validity of the OD matrix. The OD matrices assume traffic analysis zones (TAZs) while in our simulations trips start on links. We converted traffic analysis zones to links by the following heuristic. First, the geographic location of the zone is found via the geographical coordinate of its centroid given by the database. Next, a circle with radius 3 km is drawn around the centroid. Finally, each link starting within this circle is now a possible starting link for the trips. One of these links is randomly selected and the trip start or end is assigned. This led to a list of approximately 5 million trips, or about 1 million trips between 6 a.m. and 9 a.m.. 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.

Since an agent should keep more than one plan during the iteration process, the memory requirements of one million agents exceeded the available memory. So we restricted our interests to the Zurich Area only. This was done with the following steps: (i) all trips are routed using free flow travel times; (ii) we define the area of interest as a circle of 26 km radius around the center ("Bellevue") of Zurich City, and (iii) each trip that does not cross this area is removed. This results in 260,275 trips between 6 a.m. and 9 a.m.. All trips are now identified with an agent. The "origin" location for the morning trip is assigned to the home activity, and the "destination" location is assigned to the work activity. The end time of the home activity is set to the departure time of the original trip. The daily patterns "home-work" are then extended to the "home-work-home" pattern, where the two homes are at the same location. The duration of the "work" activity is set to 8 hours, with no fixed activity end time. At the end we get 260,275 agents that have an initial day plan.

## TRAFFIC COUNT DATA

There are about 230 automatic counting stations registered with the Swiss Federal Roads Authority (Bundesamt fuer Strassen). Of those, we had hourly traffic count data for 75 stations. A total of 33 of these could be located unequivocally on our network. Unfortunately there are only 6 useful bi-directional counting stations left in the Zurich area, implying we can compare 12 links with reality.

## SIMULATION PARAMETERS

The maximum number of plans per agent, $N$, was set to 5 plans. The value of the empirical constant $\beta$ used to convert plan scores to selection probabilities is $2.0/Euro$. We use the following values for the marginal utilities of the utility function used for calculating scores:

$$\beta_{perf} = +6 Euro/h,\ \beta_{travel} = -6 Euro/h \text{ and } \beta_{late} = -18 Euro/h$$

Although it is not obvious at first glance, these values mirror the standard values of the Vickrey scenario (Arnott *et al.*, 1993): An agent that arrives early to an activity must wait for the activity to start. During this time, the agent cannot perform *any* activity and therefore forgoes the $\beta_{perf} = +6 Euro/h$ that it could accumulate instead (opportunity cost). An agent that travels foregoes the same amount, *plus* a loss of $6 Euro/h$ for travelling. And finally, an agent that arrives late receives a penalty of $18 Euro$ per hour late, but is not losing (or gaining) any time elsewhere by being late. We only look at daily activity chains that consist of one home and one work activity. The "typical" times were set to $t_h^* = 16\ hours$ and $t_w^* = 8\ hours$. With these assumptions, the maximum score is $120\ Euro$ ($60\ Euro$ per activity). For the work activity a starting time window is defined between 7:08 a.m. and 8:52 a.m.. The blending factor $\alpha$ is set to 0.1. This is a useful compromise between zero learning and overreaction. We expect that changes in $\alpha$ will mostly affect the speed of relaxation; this may be a topic of future research.

## RESULTS

### Overview

We present the results of four different setups, which result from two different initial conditions and from using time re-planning or not. The two initial conditions are:

*Initial departure times given externally*: Here, the activity end times from the home activity are generated as described earlier. When the home activity ends, agents immediately depart and drive to work, where they stay for 8 hours, and then return. We will call the two setups where agents initially use externally defined times *times-routes-initial-times-extern* and *routes-only-initial-times-extern* when times re-planning is enabled and disabled, respectively.

*All agents depart home at 6 a.m.*: Once departed, agents drive to work, where they work for 8 hours, and then return. These initial conditions are used to have a scenario where the simulation starts with a clearly implausible situation. The question that is tested is whether it will recover to a realistic solution by itself. We will call the two setups where all agents depart at 6 a.m. *times-routes-initial-times-all6a.m.* and *routes-only-initial-times-all6a.m.* when times re-planning is enabled and disabled, respectively.

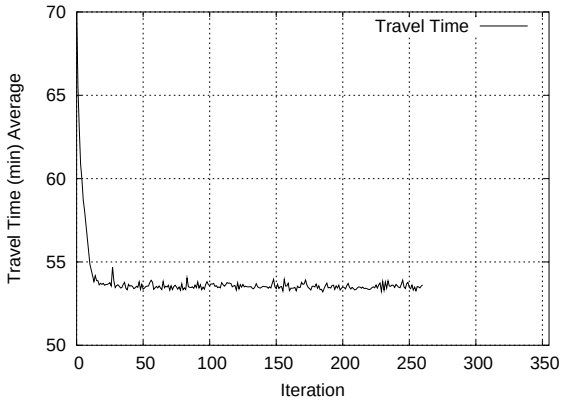
Note that when times re-planning is disabled, only 10 % of agents perform route re-planning, but when it is enabled, a total of 20 % of agents perform route re-planning, with half of those also performing times re-planning. We compare the results with the following indicators: (i) *Average travel time*: The average travel time across all agents plans for each iteration; (ii) *Average score*: The average score across all agents for each iteration; (iii) *Departure and arrival time histograms*: The number of agents that arrive/depart from an activity over time during a certain iteration; (iv) *Traffic count data comparison*: Mean bias and error of the simulations compared to the counting data described above.

**Initial Plans with Externally Defined Departure Times**

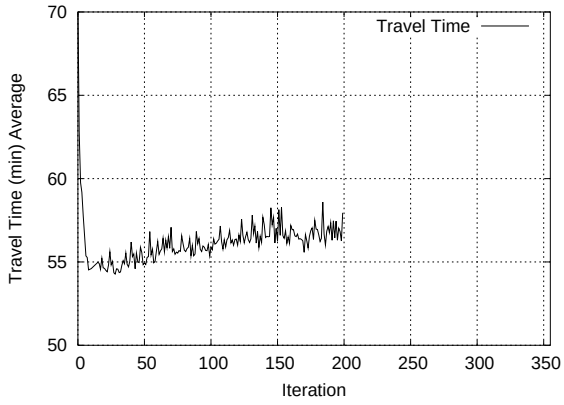
This setup tests whether or not the learning, once time re-planning is switched on, drifts away from the time structure given by the external data. Since these initial plans are based on realistic time distributions, one would assume that the time re-planning will not affect the result that much. Re-routing alone should decrease the average travel time and congestion. Figure 2 compares the average travel times over the iterations. The routes-only iteration (Figure 2a) quickly gets to a stable result because re-routing is the only part, which has to be optimized. The small fluctuations are due to the fact that some percentage of the agents always re-plans, and that the traffic flow simulation is stochastic.

The iterations where time re-planning is switched on (Figure 2b) behave in a similar way, but the average travel time is slightly higher than routes-only and also it fluctuates more. However, the scores of the times-routes setup are not worse than the scores of the routes-only setup. This indicates that the agents are “trading off” travel time for other parts of their utility. In other words,

by adjusting their activity times (i.e., the times they make their trips) they make up for the fact that trips are longer by arriving at a more suitable time to work. The higher fluctuations can be attributed to the fact that there are now two re-planning parts, which have to be optimized.

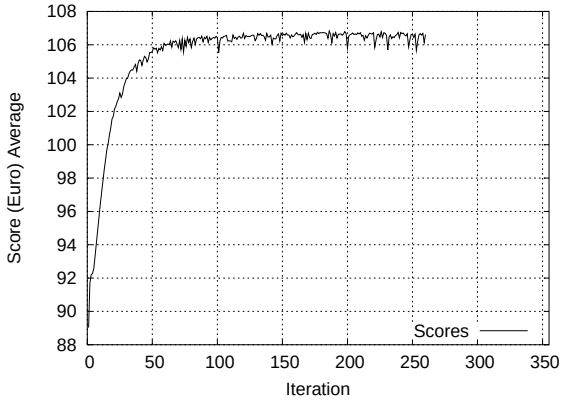



(a) Average Travel Times per Iteration for Routes-Only-Initial-Times-Extern Setup

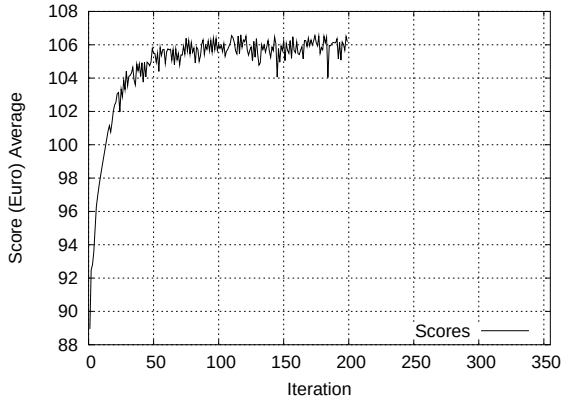



(b) Average Travel Times per Iteration for Times-Routes-Initial-Times-Extern Setup

**Figure 2**
**Average Travel Times of Routes-Only-Initial-Times-Extern and Times-Routes-Initial-Times-Extern**

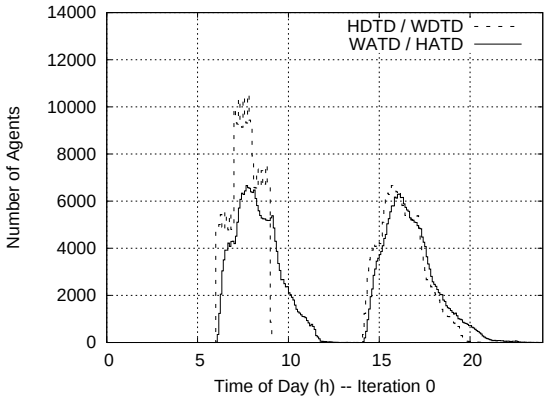



(a) Average Scores per Iteration for Routes-Only-Initial-Times-Extern Setup

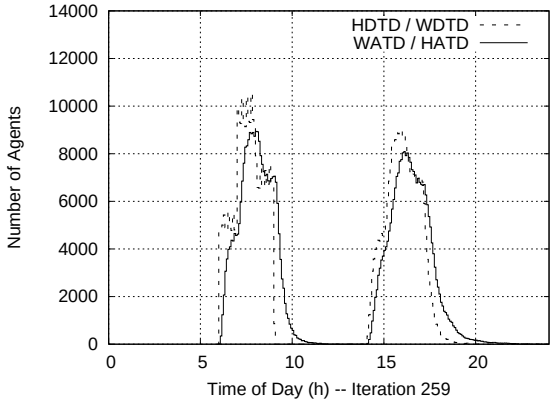



(b) Average Scores per Iteration for Times-Routes-Initial-Times-Extern Setup

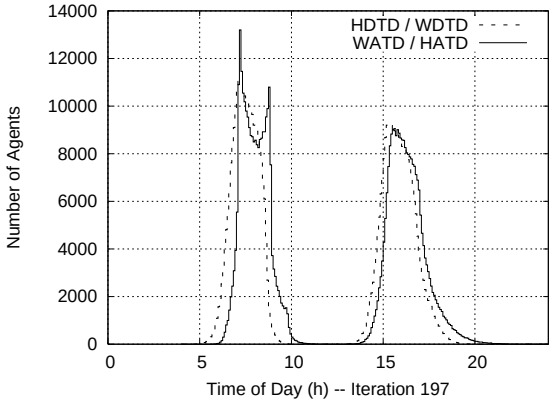
**Figure 3**
**Average Scores of Routes-Only-Initial-Times-Extern and Times-Routes-Initial-Times-Extern**




(a) Arrival and Departure Histograms (5 min Time Bins) of Iteration 0 with "Plausible" Initial Activity Times




(b) Arrival and Departure Histograms (5 min Time Bins) of Iteration 260 with Time Re-Planning Switched off




(c) Arrival and Departure Histograms (5 min Time Bins) of Iteration 200 with Time Re-Planning Switched on

**Figure 4**
**Arrival and Departure Histograms when the Initial Plans have "Plausible" Departure Times**

Figure 3 shows the scores for each iteration of both setups. They are once more similar to each other, and once more the routes-only setup (Figure 3a) shows less fluctuation than the setup with time re-planning (Figure 3b). The reason is the same as described above. Comparing to Figure 2, one can see that in both setups, the average scores relax considerably more slowly than the average travel times. This is due to the score averaging in the agent database.

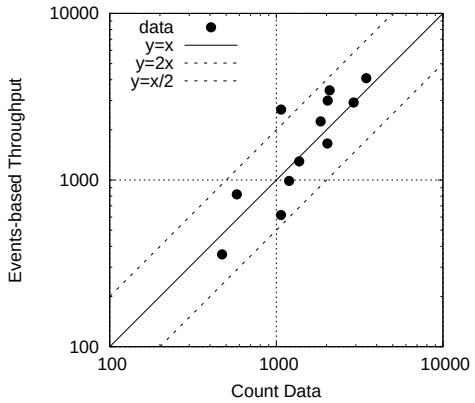
The histograms (Figure 4)show how the re-planning affects the agents. Starting with the same configuration (Figure 4a), the routes-only iteration only tries to minimize travel times, so that the periods of arrivals decreases (see bold graph of Figure 4b), while departure from home stays the same (see dotted graph of Figure 4b). Switching on time re-planning changes also the dotted graph (see Figure 4c). The two peaks of the arrival (bold) graph are at 7:08 a.m. and 8:52 a.m., which is the border of the time window we defined for these scenarios.

**Table 1**
**Bias and Error of Routes-Only-Initial-Times-Extern and Times-Routes-Initial-Times-Extern Compared to Field Data at 7-8 a.m.**

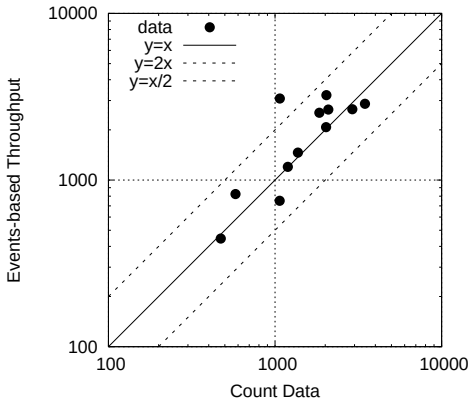
| | Initial-Times-Extern | |
|---|---|---|
| **Mean / Bias** | **Routes-Only 7-8 a.m.** | **Times-Routes 7-8 a.m.** |
| Mean Absolute Bias: | +331.403 | +306.320 |
| Mean Relative. Bias: | +19,6 % | +25.3 % |
| Mean Absolute Error: | 533.553 | 503.768 |
| Mean Relative Error: | 37.5 % | 35.4 % |

The reason for that is the fact that agents, which are too late or too early at work try to "squeeze" into this time window. Once they are inside the time window they will more or less stay at this plan if they succeeded. Since an "optimal" plan for an agent is still to have short travel times, more and more agents try to arrive earlier in the defined time window. That is why the left peak is higher than the right one.

Finally, we look at the traffic count data. Figure 5 shows the relations of the two setups and the real data given by the already mentioned 12 links. As expected, the two results do not differ very much, and they are comparable to reality. Also the quantitative measures of bias and errors are similar (Table 1).




(a) Traffic Count Data vs. 260th Iteration of Routes-Only-Initial-Times-Extern Setup at 7-8 a.m.



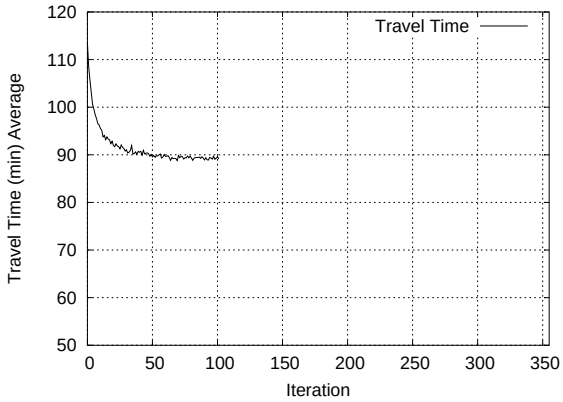

(b) Traffic Count Data vs. 200th Iteration of Times-Routes-Initial-Times-Extern Setup at 7-8 a.m.

**Figure 5**
**Initial Plans with Externally Defined Departure Times: Comparison to Traffic Count Data**

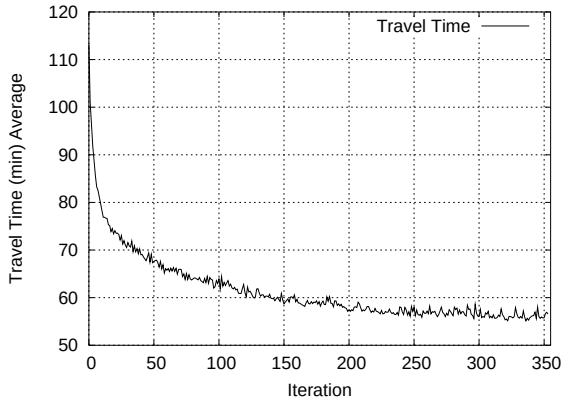
**Initial Plans with Departure Time at 6 a.m. for all Agents**

The previous section demonstrated that the results both with respect to the time structure and with respect to validation do not (at least) become worse when time re-planning is switched on. However, the initial condition was still based on the externally given time structure. The experiments in this section will test in how far a realistic time structure can be generated even when starting from a clearly implausible initial condition. For this purpose, all initial plans will be modified so that all agents initially depart at 6 a.m.. Apart from that, the initial plans are the same as before.

Figure 6 shows again the average of travel times for both setups. We see that this time, the routes-only setup decreases travel time more slowly than before because it is harder to avoid congestion when all agents start travelling at the same time. Of course, at the end the average travel time will be higher. With time re-planning switched on, average travel times decrease rather quickly, because agents are now allowed to change their departure time, too. Also average scores without time re-planning (Figure 7a) show only little improvement. Only optimizing routes does not help very much because a major part of the agents will then arrive at work too early which does not increase scores (Figure 8b). When the time re-planning module is also switched on, agents are now able to have short travel times and still arrive at work within the given time window. Figure 7b shows that the average score slowly increases to the same level as in Figure 3b.

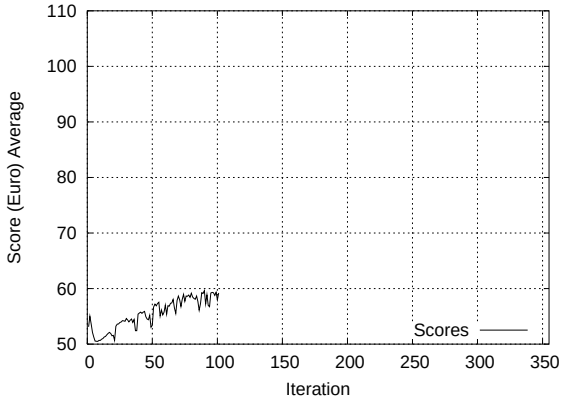



(a) Average Travel Times per Iteration for Routes-Only-Initial-Times-All6a.m. Setup

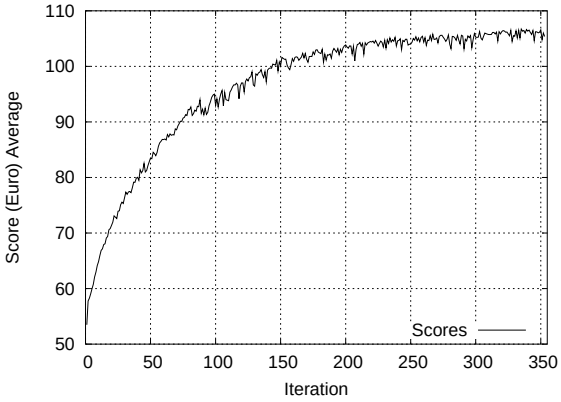



(b) Average Travel Times per Iteration for Times-Routes-Initial-Times-All6a.m. Setup

**Figure 6**
**Average Travel Times of Routes-Only-Initial-Times-All6a.m. and Times-Routes-Initial-Times-All6a.m.**

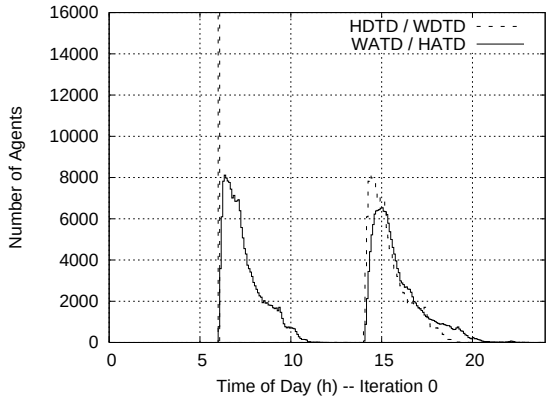



(a) Average Scores per Iteration for Routes-Only-Initial-Times-All6a.m. Setup

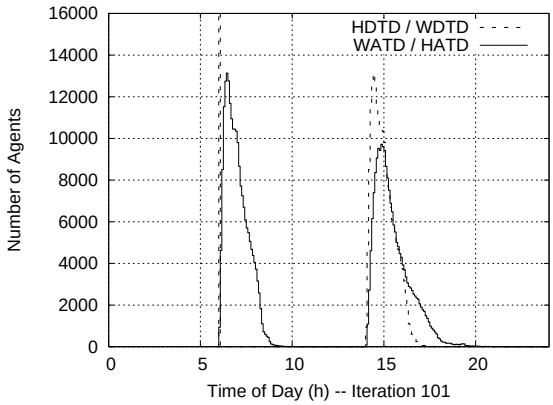



(b) Average Scores per Iteration for Times-Routes-Initial-Times-All6a.m. Setup

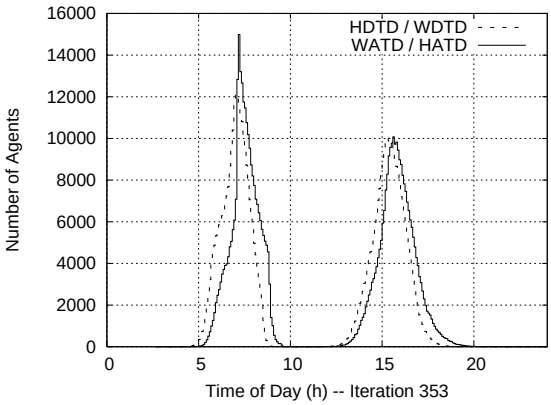
**Figure 7**
**Average Scores of Routes-Only-Initial-Times-All6a.m. and Times-Routes-Initial-Times-all6a.m.**




(a) Arrival and Departure Histograms (5 min Time Bins) of Iteration 0 with Initial Departure Ttime 6 a.m.

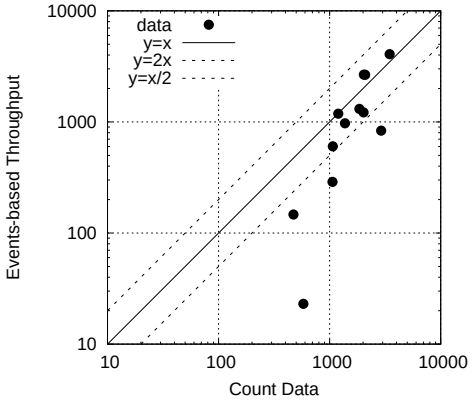



(b) Arrival and Departure Histograms (5 min Time Bins) of Iteration 100 with Time Re-Planning Switched off

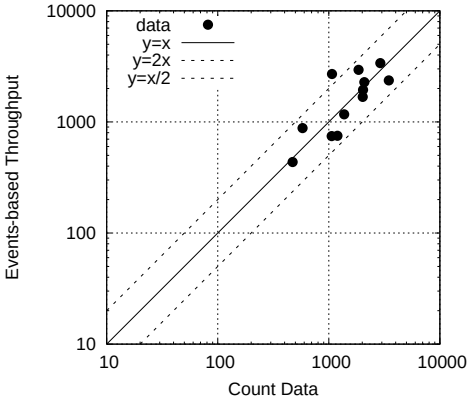



(c) Arrival and Departure Histograms (5 min Time Bins) of Iteration 350 with Time Re-Planning

**Figure 8**
**Arrival and Departure Histograms when in the Initial Plans Everybody Departs at 6 a.m.**




(a) Traffic Count Data vs. 100th Iteration of Routes-Only-Initial-Times-All6a.m. Setup at 7-8 a.m.




(b) Traffic Count Data vs. 350th Iteration of Times-Routes-Initial-Times-All6a.m. Setup at 7-8 a.m.

**Figure 9**
**Departure Time 6 a.m. Plans: Comparison to Traffic Count Data**

The histograms (Figure 8) also show those facts. There are many more people who arrive between 6 and 7 a.m. in the routes-only setup (Figure 8b) than in the times-routes setup (Figure 8c). The peak of the departure time (dotted) graph of Figure 8c moved toward the same time as shown in Figure 4c of the previous section.

Comparing the results with real word data shows a high discrepancy between the two setups. In the routes-only setup almost everybody starts too early. So it underestimates the throughput between 7 and 8 a.m. (Figure 9a). In the times-routes setup (Figure 9b), agents slowly move to more appropriate departure times which—at the end—will converge to similar results as obtained before. Of course, the calculation of the bias and the error (Table 2) now produces completely different results for the routes-only setup.

**Table 2**
**Bias and Error of Routes-Only-Initial-Times-All6a.m .**
**and Times-Routes-Initial-Times-All6a.m. Compared to Field Data at 7-8 a.m.**

| | Initial-Times-All6a.m. | |
|---|---|---|
| **Mean / Bias** | **Routes-only 7-8 a.m.** | **Times-Routes 7-8 a.m.** |
| Mean Abs. Bias: | -344.764 | +99.236 |
| Mean Rel. Bias: | -31.3 % | +12.4 % |
| Mean Abs. Error: | 644.107 | 520.256 |
| Mean Rel. Error: | 43.8 % | 36.1 % |

## COMPUTATIONAL ISSUES

*Performance*: One iteration takes on average about 102 minutes. The average duration of sub-steps of an iteration are: about 18 sec for the departure time allocation module; about 23 min for the router module, of which about 19 min is spent reading the events; about 39 min for the traffic flow simulation module, including file input and output (Cetin and Nagel, 2003); about 11 respectively 16 min for sorting events and reading processing them into scores; about 105 sec for writing the new plans; the remaining time is used for other I/O processes/bottlenecks, communication and data preparations. These times allow the calculation of 15 to 20 iterations per day.

*Disk Usage:* A complete data set generated by *one* iteration produces about 280 MB of data (when compressed). These will be kept for the first and the last 5 iterations and also for every 10th iteration. For each of the other iterations only about 40 MB are kept.

*Memory Usage:* Since we are simulating about 260,000 Agents with most 5 different plans and each of them needs about 700 Bytes of memory plus some overhead, we end up with a requirement of about 1 GB of memory. The router module also needs about 200 MB of memory. Higher resolution networks will need more memory which might become a problem in the future.

## FUTURE WORK

At present we only model the "primary" activities "home" and "work". We are working on adding "secondary" activities, such as shopping and leisure to the system. This requires the addition of two more modules: the activity pattern generator and the activity location generator. Another module we are interested in adding is a *Population generation* module, which would disaggregate demographic data to obtain individual households and individual household members, with certain characteristics, such as a street address, car ownership or household income (Beckman *et al.*, 1996; Frick, 2004). The population would not match reality, but would result in the same statistics. These modules should also be implemented as "plug-ins". We are also investigating other travel modes such as public transport or pedestrian mode.

Another issue of interest is the possibility that agents could also learn during the day. They could re-route while they are stuck in congestion, drop an activity because they are already too late, and so on. This "within day re-planning" (e.g., Axhausen, 1990; Cascetta and Cantarella, 1991) should help to improve their strategies faster than only "day-by-day re-planning", and the interaction with other entities (like traffic lights, changing traffic signs and other ITS entities) can be added to the traffic flow simulation. Within-day learning is more realistic since some types of decisions are

made on time scales much shorter than a day (Doherty and Axhausen, 1998). However, within-day re-planning is at odds with the parallel computing approach to the traffic flow simulation (Nagel and Marchal, 2003), which is the reason why it is currently not used in our project.

Simulation speedup can also be improved by elimination of performance bottlenecks. At the moment the agent database keeps track of *all* agents of the simulation. Since recalculating an agents' strategies is completely independent to other agents, it would be useful to introduce parallelism into this. These "multiple agent databases" should then be controlled by a separate module, which keeps track of the feedback. This leads us to a clear separation of "agent databases" and "feedback".

The activity time allocation module itself could be improved, too. It should recognize when agents are too early or too late, so the adaptation to a more realistic departure time should be done with fewer iterations. High resolution networks are another issue, especially if there is more precise information available about locations. The main goal will be that each agent has its home location at a street with a house number, possibly a ramp to its garage, a private pedestrian path to the next tram station, and so on. Last but not least, high resolution scenarios are indeed a computational challenge. Quite in general, more precise traffic count data is required. There is some effort to extract information of the raw data of the Kanton Zurich, which gives more precise information about local traffic situations.

## ACKNOWLEDGMENTS

The ETH-sponsored 192-CPU Beowulf cluster "Xibalba" performed most of the computations. Marc Schmitt is doing a great job maintaining the computational system. Nurhan Cetin provided the traffic flow simulation. The work also benefited from discussions with Fabrice Marchal. Michael Balmer and Bryan Raney are funded in part by the ETH project "Large scale multi-agent simulation of travel behavior and traffic flow"; Bryan Raney is also funded in part by the ETH project "A unified approach for agent-based learning with application in architecture and in transportation planning".

## REFERENCES

Arnott, R., A. De Palma and R. Lindsey (1993). A structural model of peak-period congestion: a traffic bottleneck with elastic demand. *Am. Ec. Rev.*, **83**, 161.

Avineri, E. and J.N. Prashker (2003). Sensitivity to uncertainty: need for paradigm shift. *Proc. 82nd Ann. Meet. Transpn. Res. B.,* Washington, D.C. (CD-Rom).

Axhausen, K.W. (1990a). A simultaneous simulation of activity chains. In: *New Approaches in Dynamic and Activity-based Approaches to Travel Analysis* (P. M. Jones, ed.), pp. 206–225. Avebury, Aldershot.

Axhausen, K.W. (1990b). Judging the day: a synthesis of the literature on measuring the utility of activity patterns. Working Paper 561, Transport Studies Unit, University of Oxford, Oxford.

Beckman, R.J., K.A. Baggerly and M.D. McKay (1996). Creating synthetic base-line populations. *Transpn Res. A*, **30**, 415–429.

BfS—Bundesamt fuer Statistik und Dienst fuer Gesamtverkehrsfragen (1996). Verkehrsverhalten in der Schweiz 1994. *Mikrozensus Verkehr 1994*, Bern. See also www.statistik.admin.ch/-news/archiv96/dp96036.htm.

Bottom, J.A. (2000). Consistent Anticipatory Route Guidance. PhD thesis. Massachusetts Institute of Technology, Cambridge, MA.

Cascetta, E. and C. Cantarella (1991). A day-to-day and within day dynamic stochastic assignment model. *Transpn. Res. A*, **25**, 277–291.

Cetin, N. and K. Nagel (2003). A large-scale agent-based traffic microsimulation based on queue model. *Proc. Swiss Transp. Res. Conf.,* Monte Verita, Switzerland. See also www.strc.ch.

Charypar, D. and K. Nagel (in press). Generating complete all-day activity plans with genetic algorithms. *Transpn.*

De Palma, A. and F. Marchal (2002). Real case applications of the fully dynamic METROPOLIS tool-box: an advocacy for large-scale mesoscopic transportation systems. *Netw. & Spat. Ec.*, **2**, 347–369.

Doherty, S.T. and K.W. Axhausen (1998). The development of a unified modeling framework for the household activity-travel scheduling process. In: *Verkehr und Mobilitaet* (K.-J. Beckmann, ed.), pp. 45-59. Stadt Region Land No. 66, Institut fuer Stadtbauwesen, Technical University, Aachen, Germany.

Ettema, D.F., H.J.P. Timmermans and T.A. Arentze (2004). Modelling perception updating of travel times in the context of departure time choice under ITS. *Intell. Transpn. Syst.*, **8**, 33-43.

Frick, M.A. (2004). Generating synthetic populations using IPF and Monte Carlo techniques: some new results. *Proc. Swiss Transp. Res. Conf.*, Monte Verita, Switzerland. See also www.strc.ch.

Gawron, C. (1998). An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *Int. J. Modern Phys. C*, **9**, 393-407.

Holland, J.D. (1992). *Adaptation in Natural and Artificial Systems*. Bradford Books, New York.

Jacob, R.R., M.V. Marathe and K. Nagel (1999). A computational study of routing algorithms for realistic transportation networks. *ACM J. Exp. Algorithms*, **4**/1999es/6 (electronic journal).

Kaufman, D.E., K.E. Wunderlich and R.L. Smith (1991). An Iterative Routing/Assignment Method for Anticipatory Real-Time Route Guidance. IVHS, Technical Report 91-02, University of Michigan, Department of Industrial and Operations Engineering, Ann Arbor, MI.

Nagel, K. (1995). High-Speed Microsimulations of Traffic Flow, PhD thesis, University of Cologne. See also www.inf.ethz.ch/˜nagel/papers or www.zaik.uni-koeln.de/˜paper.

Nagel, K. and F. Marchal (2003). Computational methods for multi-agent simulations of travel behavior. *Proc. Int. Assoc. Trav. Beh. Res.*, Lucerne, Switzerland. See www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.

Nagel, K., M. Strauss and M. Shubik (2004). The importance of timescales: simple models for economic markets. *Phys. A*, **340**, 668-677.

Palmer, R.G., W.B. Arthur, J.H. Holland, B. LeBaron and P. Tayler (1994). Artificial economic life: a simple model of a stockmarket. *Phys. D,* **75**, 264-274.

Raney, B., N. Cetin, A. Voellmy, M. Vrtic, K. Axhausen and K. Nagel (2003). An agent-based microsimulation model of Swiss travel: first results. *Netw. & Spat. Ec.*, **3**, 23-41.

Raney, B. and K. Nagel (2003). Truly agent-based strategy selection for transportation simulations. *Proc. 82nd Ann. Meet. Transpn. Res. B.,* Washington, D.C. (CD-Rom).

Raney, B. and K. Nagel (in press). An improved framework for large-scale multi-agent simulations of travel behavior. In: *Towards Better Performing European Transportation Systems* (P. Riedveld, B. Jourquin and K. Westin (eds.).

Timmermans, H.J.P., T.A. Arentze and D.F. Ettema (2003). Learning and adaptation behaviour: empirical evidence and modelling issues. *Proc. ITS Conf.*, Eindhoven, The Netherlands (CD-Rom).

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

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

Weiss, G. (1999). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA.