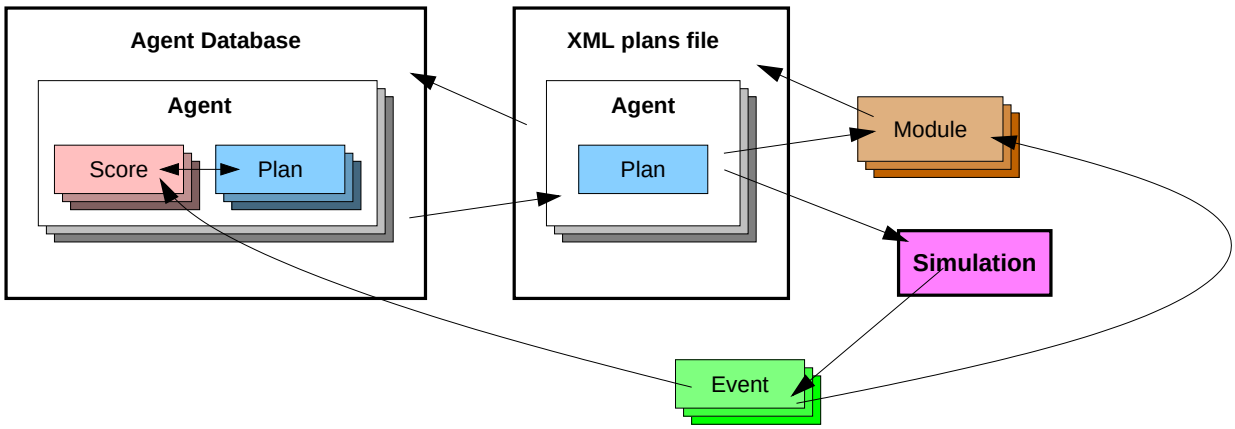


An Improved Framework for Large-Scale Multi-Agent Simulations of Travel Behavior

Bryan Raney, Dept. of Computer Science, ETH Zürich
Kai Nagel, Dept. of Computer Science, ETH Zürich

Conference paper STRC 2004

STRC | **4th Swiss Transport Research Conference**
Monte Verità / Ascona, March 25-26, 2004

An Improved Framework for Large-Scale Multi-Agent Simulations of Travel Behavior

Bryan Raney
Department of Computer Science
ETH Zürich
CH-8092 Zürich, Switzerland

Phone: 01-632 08 92
Fax: 01-632 13 74
eMail: braney@inf.ethz.ch

Kai Nagel
Department of Computer Science
ETH Zürich
CH-8092 Zürich, Switzerland

Phone: 01-632 54 27
Fax: 01-632 13 74
eMail: nagel@inf.ethz.ch

Abstract

We describe a framework for running large-scale multi-agent simulations of travel behavior. The framework represents each traveler as an individual "Agent" that makes independent decisions about its desired use of the transportation system during a typical day. An Agent keeps a record of its decisions in a "Plan." A Plan contains the Agent's schedule of activities it wants to perform during the day, including times and locations, along with the travel modes and routes it intends to utilize to travel between activities.

An Agent database gives every Agent a memory where it can store several possible Plans, as well as performance information it uses to compare how well different Plans meet its needs. Agents score a Plan's performance based on the output of the micro-simulator. The Agent database also allows Agents to periodically generate new Plans by connecting them to behavioral Modules that model the different kinds of decisions that affect an Agent's Plan. For example, one Module chooses routes, another chooses activity durations. This paper describes the design and our current implementation of this framework, plus the results of some verification scenarios.

Keywords

Multi-Agent Simulation – Transportation Planning – Simulation Framework – 4th Swiss Transport Research Conference – STRC 2004 – Monte Verità

1. Introduction

The established model for transportation planning is the four-step process, consisting of the four modules trip generation, trip distribution, modal split, and route assignment. It is well known that the traditional (static) four-step process falls short of many requirements that are desirable for modern transportation planning, for example:

1. There is no time-of-day in the static modeling approach.
2. Because there is no time-of-day, it is difficult or impossible to model any kind of time-dependent effect, such as emissions (which depend on engine temperature, which in turn depends on how long the car has been driving), or peak traffic spreading.
3. Decisions are decoupled from persons and therefore from demographic attributes.

Item 3 is, at least conceptually, easy to fix by making the first three steps of the four step process explicitly person-dependent. The most common solution to this is **activity-based demand generation (ADG)**, which is discussed at many places (e.g. Hensher and King, 2001), and implemented at some (e.g. Bowman *et al.*, 1999; Vovsha *et al.*, 2002; Jonnalagadda *et al.*, 2001). ADG typically means the following steps:

(i) Generation of a **synthetic population** by disaggregating census data into individual people. The synthetic population is a random realization of the census, that is, a census taken from the synthetic population would, within statistical limits, return the original census.

The typical data content of a synthetic population are households, which are located spatially, and which possess some attributes, such as household income, or car ownership. These households are populated with individuals, who posses additional attributes, such as gender and age.

(ii) For each individual of the synthetic population, a complete **daily activity plan** is then generated. The word “activity” refers to actions such as “being at home”, “shopping”, “working”, “being at school”, etc. Besides the activity pattern, the activity plan also contains the location of each activity, and some timing information, such as when activities are started and ended.

(iii) Each individual selects a **mode** of transportation.

It is more difficult to solve items 1 and 2, i.e. the lack of time-dependence of static assignment. The advantage of static assignment over other methods is that it has a range of mathematically proven properties, in particular the uniqueness of the solution in terms of the link volumes. Clearly, this simplifies the comparison of implementations and the interpretation of different scenarios enormously.

When making the assignment formulation dynamic (**dynamic traffic assignment, DTA**; e.g. Kaufman *et al.*, 1991; Astarita *et al.*, 2001; Friedrich *et al.*, 2000), the extent of mathematically proven properties becomes much smaller. In particular, when the dynamic formulation includes spillback (also called physical queues), then one can construct examples of non-uniqueness, i.e. there are multiple user equilibrium solutions to the same origin-destination matrix and the same network (Daganzo, 1998). In consequence, one may have to accept that DTA with spillback is in general mathematically less well-behaved than static assignment.

Conceptually, DTA can be decomposed into two components (Bottom, 2000): route generation, and **network loading**.[1] In static assignment, the network loading is done via the volume-cost function, which returns the cost of a link as a function of the trips using that link. In a dynamic context, the relationship is much more complicated, and it makes sense to look at simulation as a solution to the network loading. Simulation, as is well known, is a technique where a dynamic model is implemented on a computer, and run forward in time. Its conceptually simplest incarnation with respect to transportation planning is a representation of roads, and a way to move traffic forward along the links. Network loading models are classified according to the following criteria:

1. Resolution: Traffic can be represented by individual vehicles, but vehicles can also be aggregated into packets or cells.
2. Fidelity: The behavior of each individual entity can be more or less realistically represented.
3. Modes: The simulation can concentrate on one mode only, or can combine several modes including their interaction.
4. Time resolution/time step.

Note that at one end of this classification, one finds the traditional assignment model (resolution aggregated on link level; fidelity reduced to volume-cost-functions; car mode only; no time-dependency). Examples for more realistic, simulation-based network loading models are DYNAMIT (DYNAMIT-www, accessed 2003), DYNASMART (DYNASMART-www, accessed 2003), METROPOLIS (de Palma and Marchal, 2002), TRANSIMS (TRANSIMS www page, accessed 2003), or the queue model (Gawron, 1998a,b).

So far, this introduction discusses that demand generation can be made more realistic by moving to activities, and traffic assignment can be made more realistic by making it time-dependent and then using simulation for the network loading. Since these were discussed as separate changes, it is natural to assume that they are designed so that they are backwards compatible to the 4-step process, which means that the ADG produces origin-destination (OD) matrices as output, and the DTA takes them as input. This also means that ADG can be fed into a traditional static assignment, and DTA can take its input from the traditional demand generation.

[1] Bottom (2000) adds a third component, the guidance map, which is relevant for ITS (Intelligent Transportation Systems) applications.

However, in order to take account of the time dependency of the demand, the OD matrices generated by ADG are usually time dependent. This is a break with the backwards compatibility, in the sense that although the DTA models work with a traditional static OD matrix, it is difficult to relate the output to reality. METROPOLIS (de Palma and Marchal, 2002) is an exception to this since it takes static OD matrices as inputs and makes them time-dependent internally, using Vickrey's timing model (Arnott *et al.*, 1993).

An additional disadvantage of using OD matrices to couple ADG to DTA is that it gives up the connection to the individual persons – the same connection that was just gained in steps one to three of the 4-step process by moving to ADG. However, routing decisions can depend on individual attributes: The decision to use a toll road can depend on income; a person in need to catch an airplane may prefer a road with lower variability; etc. Also, activity chains have dependencies in the time direction – a delay in the morning may trigger changes in the afternoon – and the OD matrix severs this connection. It makes sense, therefore, to bypass OD matrices completely and to feed the complete information from the activity-based demand generation into the DTA. This means that throughout the whole process, including the DTA, the travelers are maintained as individual entities with individual attributes, and make individual decisions based on these attributes. This is what is meant by an **agent-based** or **multi-agent** approach (e.g. Ferber, 1999). And indeed, there are numerous papers related to transport that mention "agent" in their title (e.g. Arentze *et al.*, 2000; Wahle *et al.*, 2002).

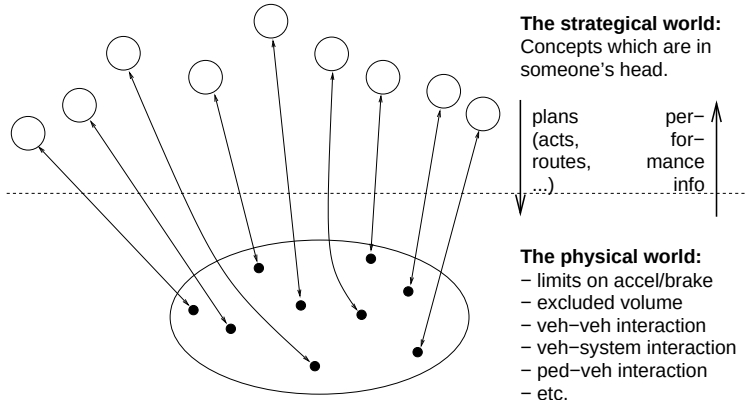
This paper concentrates on the multi-agent approach as an improvement of the complete 4-step process. The main differences against coupling ADG to DTA via OD matrices are:

- The DTA is completely agent-based, as discussed above. In particular, it is capable of feeding back agent-based information, not just link-based information.
- The ADG is included into the feedback process. Historically, systematic feedback is mostly between the route generation and the network loading; feedback to the demand generation was often done manually by the analysts. However, it has been said for a long time (e.g. Loudon *et al.*, 1997) that this process should be automated.

It is useful to conceptually differentiate between a mental and a physical layer (Fig. 1):

- The **mental layer** represents the processes that are internal to the travelers. It is sometimes also called the **strategic layer**.
- The **physical layer** represents what the travelers actually do in the physical world. In traffic, this is normally called the network loading, or the traffic (micro-)simulation. In this paper, it will be called the **mobility simulation**, to stress its total independence from the mental/strategic layer.
- There are mechanisms that couple the two layers: The attempt to execute a plan causes changes in the physical layer; inability to execute the plan as intended, for example because of congestion, feeds back into the strategic layer.

Figure 1: Physical and strategic layers of the framework.



There is, to our knowledge, no simulation package that executes this approach in its entirety. Part of the challenge is that this necessitates a large number of modules and module interfaces, which in itself is quite a challenge, in particular in view of the fact that there are few programming and/or data exchange standards in the community. Another part of the challenge is that one needs parallel computing techniques for metropolitan-size scenarios, and no established technology is available to even define a viable standard for module interaction once the simulation becomes parallel (Nagel and Marchal, 2003). Some partial packages are discussed in the following.

Dynamic traffic assignment (DTA)

There is a large number of packages that do dynamic traffic assignment (e.g. DYNAMIT-www, accessed 2003; DYNASMART-www, accessed 2003; de Palma and Marchal, 2002; Gawron, 1998b). As discussed above, these packages typically take time-dependent OD matrices as input, assign routes according to user equilibrium, and return time-dependent link travel times and other link-based information as output. Although most of these packages have individual travelers inside their model, often these are not fully developed. For example, routes are calculated by the network rather than by the agent, and agent-based output is often not available. While the latter is conceptually easy to fix, the former means that making route choice dependent on agent attributes is close to impossible. This is related to the fact that those models take OD matrices as input, which contain no demographic attributes. A direct consequence is that it will not be possible to connect an agent's performance in the DTA to the demand generation, since the OD matrices sever the connection to the individual agents; it is only possible to feed aggregated information, such as link travel times, back to the demand generation modules. In summary, although the DTA models have a large number of agent-based elements, they are not fully agent-based.

TRANSIMS

TRANSIMS (TRANSIMS www page, accessed 2003) indeed replaces the complete four-step process by an agent-based approach that as its first step disaggregates census data and then works with individual travelers in all modules. In addition, TRANSIMS uses parallel computing to tackle large scale problems. The main shortcomings of TRANSIMS, in our view, are:

- Although travelers are individually identifiable throughout all modules, agent information is spread throughout the simulation system. For example, the activity file does not contain demographic information, and the route file contains neither demographic nor activity information. This makes it rather difficult to use such "higher level" information in the "lower level" modules. It also makes it possible, for example, that an agent leaves a location before it has arrived – this is possible since the agent is not a singular entity in the simulation.
- The file formats are rather inflexible. This makes it difficult to add/modify information, for example when adding new modules to the system.
- The feedback mechanism "forgets" old plans and their performance. Although it is a bit difficult to see at first glance, this means that TRANSIMS plans generation modules need to fulfill rather strong requirements: For example, the routing module ("router") does not only have to generate plausible routes, but also correct probabilities to choose between different alternatives.
- And finally, it was impossible to obtain an academic license, including source code, outside the United States. This made it impossible to improve the above aspects inside TRANSIMS.

Land use simulations

There are land use simulations, such as URBANSIM (URBANSIM-www, accessed 2003), ILUTE (Salvini and Miller, 2003), or that of Hunt *et al.* (2001), which are more or less agent-based, and which have the conceptual intention to couple to the transportation system. However, in practice this connection is not well established in those models at this point in time.

This paper will present an approach that is based on TRANSIMS but solves the above TRANSIMS shortcomings. The conceptual idea behind our approach is that we keep the agent concept consistent everywhere. The main technical improvements are in the following areas:

- The different TRANSIMS files are replaced by a single XML file format. That file contains, agent by agent, all information related to the agents, from demographic information via activities to routes.

- An agent cannot be "divided". This means in particular that the traffic simulation (network loading) executes daily plans, rather than just trips.
- An agent data base keeps track of agents' past plans and performance of those plans.

In the next section, we describe the basic design of the framework, including the agent and plan entities, the agent database, plan scoring, and iterations between the parts of the framework. Section 3 goes on to describe our current implementation of this framework, in particular introducing the XML data format that is one of the cornerstones of this framework. This is followed by a description, in Sec. 4, of the set-up of our case study for verifying the operation of this framework, including the specific behaviors of our mobility simulator, and route and time choice modules. This section also describes the utility function used by the modules to score plans, and the transportation scenario we have executed. Section 5 then explains the results obtained by our framework for this case study. Finally, we end with a discussion and summary.

2. Framework Design

2.1 The Core: Agents and Plans

The framework models the travel behavior of a population of people living in a given geographical region (e.g. town or metropolitan area) as they go about their lives during a certain, often repeated period. The arguably most typical example is a 24-hour weekday, but also other days (such as Sundays) or longer periods (such as complete weeks) can be modeled as long as there is some repetition from one period to the next, i.e. as long as there is something "typical" about the period.

During that period the people carry out *activities*, such as sleeping, working, shopping, eating, etc., at various locations in the region, and utilize the transportation system of that region to travel between activities at different locations. Thus, their activities motivate their travel choices. Also, the limitations caused by the transportation system, such as limited accessibility or congestion, affect their activity choices.

The framework represents each person as an individual entity, called an *agent*. Each agent makes independent decisions about which activities to perform during the day, where and when to perform them, and what travel modes and routes to take to travel between activities. In our system, each individual agent is represented as a simplified classifier system (Holland, 1992): Each agent has a collection of plans; each plan has a score which is updated after the plan was used; and the choice between plans is done according to their score. The main simplification when compared to a classifier system is that, at this point, our plans are not conditional on the environment. See for example Fig. 2.

Specifically, a plan contains the agent's intended schedule of activities for the day, and the travel *legs* connecting the activities. An activity schedule specifies the following information for each activity:

Figure 2: Example of information contained in the framework. It keeps all information contained within the agents' plans, with the addition of scores and other bookkeeping information.

Agent	Plan Num.	In-Use	Score	Plan
1	1	false	462.2	sleep until 7:00 AM; go to work at W1 for 8 h; go shopping at S1 for 30 min, ...
1	2	true	*undefined*	sleep until 8:00 AM; go to work at W1 for 7.5 h; drink beer at B1 for 1 h, ...
2	1	true	1047.8	sleep until 6:30 AM; take kid to kindergarten at K3; go to work at W2 for 6 h, ...
⋮	⋮	⋮	⋮	⋮

- type: what the agent wants to do (e.g. home, work, shopping, leisure)
- location: x/y-coordinates within the simulated region and/or the network link id (i.e. street)
- timing information: how much time the agent wants to spend at the activity after arriving at its location, and/or absolute ending time of the activity. More detail of this is discussed in Sec. 2.3.

An plan also contains leg information for each pair of consecutive activities. Each leg contains the following information:

- mode: what type of transportation to use
- travel time: estimated duration of trip
- departure time: estimate of what time the trip will begin
- route (certain modes only): the list of network nodes to traverse to get from the previous activity to the next
- (other mode-specific information)

The entity containing all agents' plans is called the **agent database**. Prototypes of the agent database can be found in (Nagel, 1994/95, 1996) and in (Raney and Nagel, in press). Nagel (1994/95, 1996) describes a system where agents remember individual scores for different plans, but the plans were the same for different agents and the implementation was for demonstration purposes only. Raney and Nagel (in press) describe prototypes of the implementation also described in this paper, but concentrate on a completely different version. That version was not general enough to go beyond dynamic traffic assignment.

2.2 Iterations

Above, it was stated that a plan's score is updated after the plan is used. This corresponds to the dichotomy between the mental and a physical layer mentioned earlier. For a single agent, this simply means that the agent keeps repeating the following steps:

1. Select a plan.
2. Execute it and record the performance.
3. Update the score of the plan based on the performance.

In addition, from time to time plans that have a low score are replaced by new plans (see Sec. 2.4). Note that "execution of a plan" means that it is submitted to the physical layer for execution. In our case, the physical layer is the mobility simulation. For a multi-agent system, one has in addition that all agents learn simultaneously. This means that a score for a plan is not necessarily stable over the iterations; in consequence, an agent needs to re-evaluate previously bad plans from time to time in order to check if the scores have improved.

The above is a generic design that should work for arbitrary multi-agent simulations. Some remarks are in order:

- It is important to have separate representations of the mental/internal and the physical/external processes. Sometimes, for example in robosoccer (e.g. Kim, 1997), it is possible that the physical layer is a real-world system, and then part of the challenge is to formulate plans such that they can indeed successfully execute in the physical world. For transportation simulations, a true physical model world will in general not be possible; instead, there will be a simulation of the physical world. However, plans should be designed in a way that they could also execute in the real world.
- The execution of the plan in the physical system can also be seen as interaction of the agent with its external environment and other agents, collecting "sensory" input about its experiences, which it then uses to update its mental state (i.e. learn).
- Inside the framework, it would be possible to make plans conditional. For example, there could be a separate plan to follow if the agent got delayed on the way to work. Activity durations (Sec. 2.3) are a (small) example of conditional behavior.
- A plan can be seen as a simple strategy from game theory. For a Nash equilibrium in game theory, an agent attempts to find a strategy that he/she cannot unilaterally improve.
- The framework as described so far is best suited for day-to-day (or more generally period-to-period) replanning. It can, however, be extended to within-day replanning. When doing this, one is confronted with some conceptual and some computational problems. These will be discussed in Sec. 6.

2.3 Conditional plans and activity durations

Some diligence is necessary in the treatment of activity durations. In principle, a daily plan is entirely defined by the given sequence of activities, the departure time from the first activity, and the durations of all subsequent activities. This can, however, lead to very implausible behavior: An agent can, for example, shop beyond the closing time of the shop, or remain in the movie theater beyond the end of the movie. In principle, an agent should not plan to do this; it may, however, happen because of unexpected delays earlier in the day. Since this is so grossly implausible, a first step toward conditional plans/strategies was implemented in our framework: Both the activity duration and the activity ending time can be specified, and the ending time takes priority over the duration if the agent's arrival time plus the duration would cause the agent to stay past the ending time. For example, say an agent wants to shop for 30 minutes starting at 17:30, knowing the selected shop closes at 18:00 (specified as the end time of the activity). If the agent arrives late to the shop, at say 17:45, the 30 minute duration would put the end of the activity at 18:15, which is after the ending time. In this case the ending time takes precedence and the agent departs at 18:00 instead. However, if the agent arrives early, at say 17:15, it still stays for 30 minutes and leaves at 17:45 instead of waiting until 18:00.

There is in fact a similar problem with the start of an activity: Assume that a shop opens at 8am, and the agent wants to arrive exactly at 8am and shop for 10 min. Now assume that the agent arrives 15 min early. The plausible thing to do would be to wait the 15 min and then shop for 10 min. Our current implementation will, however, let the agent wait for 10 min and then let him travel to the next activity location. This nonsensical behavior will probably need to be modified in future versions.

2.4 Generation of strategies/plans

The concept as described so far mostly discusses how agents maintain and use a larger number of plans; there is little discussion of how plans are generated. This is intentional since the precise methods of how strategies are generated does not matter to the overall design. One can for example use constructive algorithms (which construct plans from scratch), mutation (which locally modifies existing plans), crossover (which generates new plans by combining existing ones), etc. In particular, one can use externally existing programs that use the right type of input and generate the right type of output. The only two conceptual requirements are:

- The external module needs to "think" in terms of agents. For example, an external route generation module needs to generate a path for a specific agent between two locations, rather than, say, a shortest path tree.
- There needs to be some commonality between what the external module attempts to achieve, and how the scoring is done by the agent database. The system will work as long as *some* of the plans generated by external modules are "good" in the sense of the scoring function of the agent database. This is a considerably weaker design requirement for external mod-

ules than TRANSIMS has, where *all* of the plans generated by external modules need to be "good". This is necessary in TRANSIMS because any previous "good" (or "bad") plan an agent knows about is overwritten by the new plan generated by the module.

2.5 Scores

The agent database needs a scoring function in order to give scores to plans that were executed. The primary candidate for the scoring function is the traditional utility function, but any plausible scoring function, for example one using aspect theory (e.g. Avineri and Prashker, 2003), can be used. An example of a utility-based scoring function will be presented in Sec. 4.7. As mentioned before, there needs to be some overlap between what the external modules compute and what the scoring function optimizes.

3. Implementation of the framework

When implementing the above concept, one needs to make some decisions about the technologies to use. In our case, the most important criterion was that large scale scenarios (several millions of agents) should be feasible, followed by the desire for flexibility and interoperability.

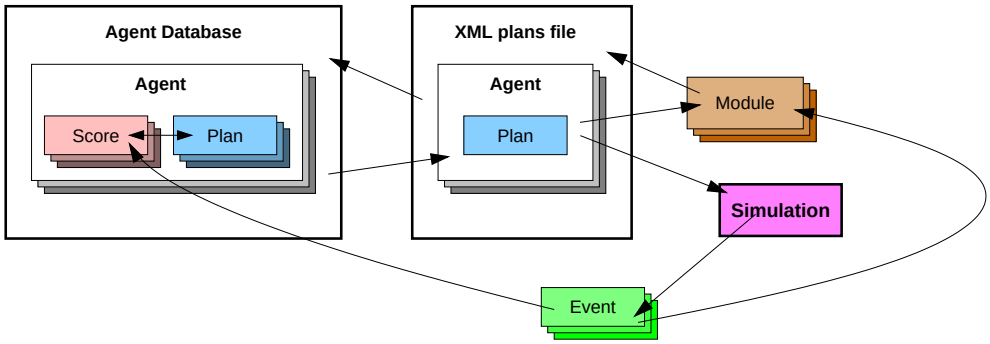
3.1 Agent database and external modules

The framework's strategic/mental layer provides the mental state of the agents and allows them to learn about their environment and make decisions about their behavior. We divide this layer into a central *agent database* and several behavioral *modules* that model the different kinds of decisions that affect an agent's plan. For example, one module chooses activity durations, and another chooses routes. Figure 3 graphically depicts the relationships and interactions between these components. The agent database provides each agent with part of its mental state, and with a high-level decision-making ability. The modules provide the rest of the mental state and more detailed decision making abilities.

3.2 Implementation of the agent database

The task of the agent database is to maintain, for each agent in the simulation, some number of plans plus their scores, to select plans according to their scores, to add new plans, and to remove plans with a bad performance. This looks like a standard data base, and in fact our first prototype was implemented in MySQL, a public domain relational database (Raney and Nagel, in press). A standard relational database is, however, not well suited to data that has hierarchies of variable-length objects. In our case, we have a large number of agents, each of which has a variable

Figure 3: Components and data flow in the framework



number of plans, each of which has a variable number of activities/legs, each of which has a route description of variable length. Since plans for a particular agent are added/removed one by one, this means that the plans of an agent are spread out in memory within the database, resulting in slow performance. In addition, the relational database approach is awkward to use, since once more agent information is not in one place.

In fact, one would need an object-oriented database, rather than a standard relational database. However, object-oriented databases are slow, which is a direct consequence of the problem to insert variable-length objects into linear memory. On the other hand, for our purposes many properties of databases, such as an always consistent state also in under crashes, are not needed. It is therefore tempting to implement the agent database completely in software. Because of performance reasons, a decision for C++ was made, and the STL (Standard Template Library) was heavily used. This allows the program to implement a `Person` class, which contains one or more `Plan` classes. Each plan contains a sequence of activities and legs, and each leg contains the description of the route. Since the STL is used, it is straightforward to, say, add or remove a plan to or from a person. Also, since the whole agent database is written in C++, it is straightforward to do computations such as plans selection based on a logit model. The number of plans that an agent database in software can hold is limited by the memory that a single process can address. In a 32-bit architecture, this number is 2 GByte. Since in our current implementation one plan needs about 0.5 KByte, our current implementation can hold about 1 million agents with a maximum of 4 plans each.

3.3 XML plans

The agent database needs to communicate with external strategy generation modules, and to send plans to the mobility simulation (Fig. 3). All communication is done by using exactly the same plans format. This format uses XML; an example is in Fig. 4. As one can see, the format is rather intuitive; this is in stark contrast to the TRANSIMS files. However, the main advantage of XML

Figure 4: A typical plan in XML. This agent, id 393241, leaves home (on link 5834) at 7 AM, and drives to work via a 4-node route (5 links) which it expects to take 25 minutes to traverse. The agent stays at work for 9 hours, then drives home again via a 2-node route. (The "100" on the x and y coordinate labels refer to the 100x100 meter blocks of census information. We do not know coordinates more accurately than that resolution.)

```
<person id="393241" income="50000">
  <plan>
    <act type="h" end_time="07:00" x100="697150" y100="232790" link="5834" />
    <leg mode="car" dept_time="07:00" trav_time="00:25">
      <route>1932 1933 1934 1947</route>
    </leg>
    <act type="w" dur="09:00" x100="700650" y100="233980" link="5844" />
    <leg mode="car" dept_time="16:25" trav_time="00:14">
      <route>1934 1933</route>
    </leg>
    <act type="h" x100="697150" y100="232790" link="5834" />
  </plan>
</person>
```

is its extensibility. That is, one can add fields to the format without breaking existing parsers. In particular, one can add fields only to a subset of agents, for example a format to describe a conditional strategy (Sec. 2.3). Such extensions would be very hard to do with TRANSIMS. It is important to note that the principal units of description are "agents" and "plans". Any external module using the same principal units will be able to communicate with our system. Somewhat unexpectedly, file size is less of an issue with XML than expected. When compressed, XML files have about the same size as TRANSIMS files with the same information.

3.4 Events

A question remains of how to feed performance information from the mobility simulation back to the strategic modules (Fig. 3). Our current solution is that the entire output of the physical simulation consists of events which are output directly when they happen. For example, a traveler can depart, can enter/leave a link, etc. That is, the simulation of the physical system performs *no* data aggregation; this is done by the other modules themselves.

At this point, we are still investigating if events should be in plain text or in XML format; there are some performance advantages to the former, but in the long run this will probably be outweighed by the flexibility advantages of the latter. An XML events format roughly looks as follows

```
<event time=".." type="leave_activity" agent_id=".." location=".."/>
```

Note that such a line is generated separately for each event.

The agent database, for example, will read through the events information and register, for each

agent, events that are necessary to compute the score. Since at this point the score depends on activity arrival and departure times only (see Sec. 4.7), these are the only events that the agent database will consider. In contrast, the router will read through the events and look for link entering/leaving events. If an agent enters a link, the router will store that information somewhere. If an agent leaves a link, the router will search for the corresponding link enter event, compute the link travel time, and enter that into some averaging mechanism for the link.

The advantage of events is that they are very easy to implement into the simulation of the physical system. In contrast, any data aggregation inside the simulation of the physical system in our experience is a continuous source of errors. This has to do with the fact that the team that writes the simulation is not truly interested in correct aggregation: Their main tool to check simulation correctness is the visual impression (and maybe some traffic flow considerations). In contrast, the team that is responsible for, say, the router or the agent database has a much higher interest in the correctness of the aggregation, since without that their module will not function. In our experience, seemingly trivial aspects such as this are rather important for the long-term robustness of the system.

3.5 Calling sequence

The modules need to be called in a certain sequence in order to make the system run. For example, choosing new activity locations will necessitate new routes to and from the changed activities, so the route planning module should be called sometime after the activity location module is called. But routes and activity times do not (strictly) depend on each other, so it would be possible to make calls to either the activity time choice module or the router without calling the other one, or call them both in an arbitrary order.

At this point, let us assume that we treat period-to-period replanning only, and that each period corresponds to a day. Within-period replanning will be shortly discussed in Sec. 6. Let us assume further that the list of available modules is known, as well as the dependencies between them, and that the dependencies can be fulfilled without calling modules in a "circular" order. Let us also assume that there is some initial plans file, in which each agent is contained, and each agent has exactly one completely specified plan. Such initial plans files can be generated with variations of the methods discussed in this paper, but the system is easier to explain if one assumes the file is already there. Finally, let us assume that the mobility simulation was run based on the initial plans file, and that it has written events to a file. This initial condition is now followed by many iterations, each composed of the following sequence of actions:

1. The agent database reads the events, interprets them, and updates the scores of each used plan.

2. The agent database, based on some behavioral model, and the information about what dependencies exist between modules, selects agents which are up for replanning on some level, and writes a corresponding plans file to disk. The level of replanning needs to be matched to an existing external module, and any other modules that this module depends

upon for plan information must have been executed previously in this iteration. The plans file written by the agent database can contain plans of arbitrary completeness, so long as all the information required by the module is available.

3. The external module is started, it reads the events information and the plans file, and updates the information that it can generate, by either overwriting existing information, or filling in blanks. Information not filled in by the module is left alone, or destroyed if it is invalidated by the new information provided by the module. For example, a route planning module may overwrite existing routes, but must not touch activity locations. On the other hand, an activity location planning module may update activity starting and/or ending locations, and must also delete those routes no longer connected to the new locations.

4. The agent database reads the new plans file, and stores the corresponding information. It then selects agents which are up for replanning on some other level, and the whole process with an external module is repeated. Within this process, the module dependencies described earlier need to be satisfied.

5. Once all module dependencies have been fulfilled and plans are completely specified, the agent database selects the plans that are to be executed in the mobility simulation. New plans, which do not have a score yet, are selected with a high probability. If an agent has not received a new plan, the agent selects between its existing plans, for example with a logit model. A specific version is discussed in Sec. 4.4. The selected plans are written to a file.

6. The mobility simulation is run based on the last plans file, and outputs events, as before.

As said before, this sequence denotes one iteration; many such iterations are run. To improve performance, the agent database stays alive throughout the whole process. This has the advantage that the several GByte of data that the agent database has stored need not be written to file during the iterations.

3.6 Specification of external strategy modules

The specifications of an external strategy module are perhaps already clear at this point. The minimum requirements are:

- The external module reads an arbitrarily complete plans file.
- If the external module reacts to agent or system performance, then it reads the events file.
- The external module writes out an updated plans file, where any invalid information has been deleted.
- The external module needs to be reasonably fast; running it on 10% of all agents should not take more than about one hour.

Note that this specification leaves the internal functioning completely to the module. In particular, the module is free to start anew with each iteration, or to accumulate information over all iterations. An example for the former is a route generator that uses link travel times from the last iteration; an example for the latter is a mental map that is built successively over the iterations.

3.7 Specification of the mobility simulation

The specifications of the mobility simulation are perhaps also already clear at this point. The minimum requirements are:

- The mobility simulation reads a complete plans file, with one plan per agent. Plans should be executed as chains, i.e. an agent can only depart *after* it has arrived at a location.
- The mobility simulation executes all those plans simultaneously.
- The mobility simulation writes events information to a file.
- There needs to be a mechanism to deal with all modes of transportation (see below).
- The mobility simulation needs to be reasonably fast; running the whole scenario once should not take more than about one hour.

In our experience, these specifications are not difficult to fulfill. They are, however, a significant departure from the way in which most current mobility simulations are written: They read OD matrices instead of plans, and they write link performance information instead of events. Writing events instead of or in addition to link performance information is relatively easy to implement. In contrast, making the simulation follow pre-specified plans sometimes necessitates a major implementation change. That change corresponds to the fact that in the agent-based approach all information is stored in the agent, whereas in many existing approaches most of the information (such as shortest path trees) is stored in the network. And in addition, conditional plans files, such as discussed in Sec. 2.3, may make the simulation logic more demanding in the future.

Furthermore, future versions will necessitate a consistent way to deal with travel in different modes. It is clear that, in order to execute traffic with different modes, the use of these modes needs to be planned by the agent database and its external modules. However, as a simplification one could just assume that the execution follows exactly the plan – this would correspond to a system without congestion, without unexpected variability, etc. In that case, there are two options:

- The agent database itself takes care of modes that the mobility simulation cannot execute. That is, they are just not included into the plans file. – The main disadvantage of this is that no corresponding events would exist, making, say, the consistent build-up of a mental map more difficult. For that reason, the following solution is preferred.

- The mobility simulation "fakes" the execution of unknown modes. For example, let us assume that a plans file has the following information:

```
<plan>
    <act type="home" location="ab" .../>
    <leg mode="walk" duration="20min">
        <route ... />
    </leg>
    <act type="work" location="cd" .../>
</plan>
```

  A simulation that can simulate the walk mode would let the agent walk along the specified route. A simulation that cannot simulate the walk mode would just assume that the walk takes 20 minutes, as specified in the `duration`, and move the agent to the next activity accordingly.

3.8 Scoring function

As mentioned elsewhere, the agent database needs a scoring function in order to give scores to plans that were executed. That scoring function needs to be entirely based on events information, and it needs to score the complete period (e.g. day). An example of a utility-based scoring function will be presented in Sec. 4.7.

An open problem is how to couple the scoring function used by the agent database to the scoring functions used by the external strategy generation modules. Because of stochastic effects, it is not necessary that they are completely consistent, but as mentioned before, some conceptual overlap is necessary. At this point, we solve this problem by manually defining the goals of the external modules. This is a subject of further investigation.

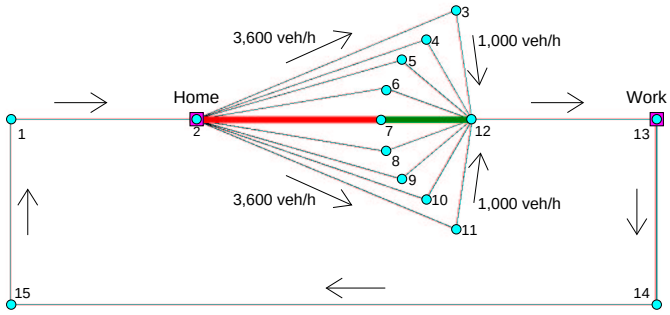
4. Verification setup

In order to test our implementation, verification scenarios were designed. Those scenarios are constructed in a way that they test the most important features of the framework. The features that are tested are: (i) capability to relax to an approximate equilibrium solution; (ii) capability to cooperate with more than one external module; (iii) capability to generate a meaningful solution even with an external module that just performs small random changes ("mutations") of existing plans.

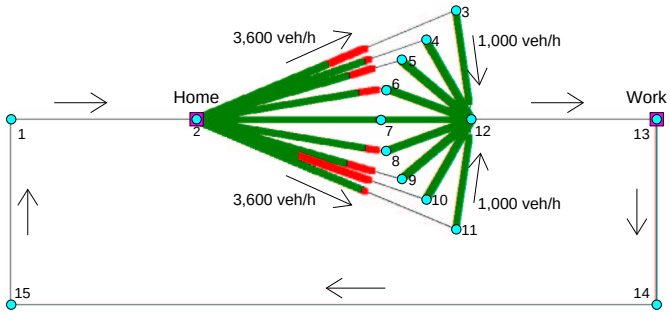
A scenario consists of: (i) the network; (ii) the initial plans file; (iii) specific implementation details. These will be treated one by one in this section.

4.1 The network

Figure 5: Diagram of the testing network overlaid onto example snapshots of the Routes Only scenario at 06:08 AM, (a) before and (b) after many iterations of route replanning. After replanning, the agents have spread onto the different available routes between home and work. Link capacities are 36,000 veh/h except where indicated (the nine route options are identical).



(a) Before replanning



(b) After replanning

The network used for this scenario can be seen in Fig. 5. It consists of a circular arrangement where in one part travelers have multiple route options. Internally, all those routes have the same lengths and capacities, so that there is no bias toward one or the other. The expectation is that in the relaxed states all those routes are used equally. All roads are uni-directional; travelers need to follow the roads clockwise.

4.2 The initial plans file

All scenarios need to start with an initial condition. In our case, we take as initial condition a plans file which contains 2'000 agents, and one fully specified plan per agent. Those plans are the same for all agents: They specify that the agents leave the "Home" location at 6:00 AM, take the middle road (through node 7) to the "Work" location, and work for eight hours. Then they take the lower part of the "circle," through nodes 14, 15, and 1, to return home. The total free-speed travel time is about 54 min, with 15 min for the trip from "Home" to "Work", and 39 min for the return trip.

4.3 The mobility simulation

As the mobility simulation we use an improved version of a so-called "queue simulation" (Gawron, 1998a). The improvements include an implementation on parallel computers, and an improved intersection dynamics, which ensures a fair sharing of the intersection capacity among incoming traffic streams (Cetin and Nagel, 2003). As pointed out elsewhere, the mobility simulation takes a plans file, executes it, and returns events information, such as when agents leave from or arrive at activities, or when agents enter/leave links.

4.4 Agent Database

As discussed elsewhere, the agent database contains all plans for all agents, and manages addition, modification, removal, and selection of plans. The precise functioning of the agent database for our test scenario is as follows:

1. The agent database may be required to limit the number of plans stored per agent to N_{plans}. If any agent ends up with more than plans N_{plans} plans, it continues to delete the plan with the worst score until the number of plans is small enough. Note that in the steps following this one, an agent may obtain a new plan, which means it will have $N_{plans} + 1$ plans while trying a new one out, but it will have only N_{plans} to choose from when selecting from old plans.

2. With probability p_{times}, agents are selected to undergo times replanning. Each of these agents selects an existing plan from memory with uniform probability (not based on score) to use as a template for the new plan. This template is immediately copied into the agent's memory, to serve as a placeholder for the eventual new plan. The selected plans are written to a file, and the time replanner (see below) is called with this file as input. It writes the resulting new plans to a separate file. The agent database reads this file and overwrites the template in each agent's memory with the corresponding plan from the file.

3. With probability p_{routes}, agents are selected to undergo route replanning. As with times replanning, each of these agents selects an existing plan as a template and copies it into its memory as a placeholder for the new plan. In addition, we set the router to be dependent on the times generated by the times replanning module, since the new leg departure times may

offer better routes than the ones currently used. Therefore, all agents that underwent times replanning select their new plans for route replanning as well. Note that no plans are copied for these agents, since the plan with the new times already serves as the template for the new routes, and overwriting it will not change the new time information. All the template plans are written to a single file, the route replanner (see below) is called, the resulting new plans are written again to file, and the agent database reads those plans and overwrites the template plans with the new ones.

4. For each agent that has multiple plans, the plan to be executed in the following mobility simulation is selected.

   - All agents that possess a new plan select that new plan.
   - A fraction p_{rnd} chooses a plan at random from their memory, using a uniform probability for each plan, and ignoring plan performance. This is done to force agents to re-evaluate existing plans from time to time, even plans with a bad score.
   - All other agents select between existing plans, with probability

     $$p_i \propto e^{\beta S_i} \ ,$$

     where S_i is the score of plan i and β is a constant (see below). This is just a standard multinomial logit model (e.g. Ben-Akiva and Lerman, 1985).

     The value of β affects how likely agents choose "non-best" plans. Higher values of β lead toward the best plan being chosen more often, while smaller values provide a more randomized choice. Since β interacts with the scaling of the scoring function, it is discussed in more detail there.
   - The selected plans are written to file; the mobility simulation runs with these plans and writes events information to a file.
   - The agent database reads the resulting events, computes corresponding scores for each agent, and adds those scores to the existing scores according to

     $$S_i = (1 - \alpha)S_i + \alpha S'_i,$$

     where S_i is the stored score for plan i, S'_i is the newly calculated score, and $\alpha (\in [0, 1])$ is a blending factor.

     This leaves the question of how plans are scored that have not been used before. Plans from the initial plan set have no history, so there is no way to estimate their scores. Upon their first use, the agent simply gives it the new score:

     $$S_i = S'_i.$$

     However, when plans are copied as templates for replanning, score information is available from other plans, which can be used to provide an estimate the initial value of S_i. This estimate can be calculated in many ways; at present we set it to the score of the best plan in the agent's memory.

For this paper, we set the above parameters to the following values:

- The maximum number of plans per agent, N_{plans}, is 6.
- The probability of an agent to perform time replanning, p_{times}, and route replanning, p_{routes}, are both 0.1.
- The probability of choosing a random plan, p_{rnd}, is 0.1.
- The plan selection constant, β, is 2/€. Section 5.4 describes what happens when we try different values of β.
- The score blending factor, α, is 0.1.

4.5 Routing module

An agent's plan must connect successive activities at different locations by travel legs that use the transportation network. Legs are described by their mode and mode-specific information. For example, a car-mode leg contains a node-by-node description of the vehicle's route through the network from the location of the previous activity to the location of the next activity. In principle legs can be selected from different mode types, but at present we only model car trips. The legs have (expected) starting times, and the router needs to be sensitive to congestion so that it can avoid using already congested links.

We currently use a router based on Dijkstra's shortest-path algorithm, but "shortness" is measured by the time it takes an agent to drive the route rather than distance. The fastest path depends on the travel times of each individual link (road segment) traversed in the route. These times depend on how congested the links are, and so they change throughout the day. This is implemented in the following way: The way a Dijkstra algorithm searches a shortest path can be interpreted as expanding, from the starting point of the leg, a network-oriented version of a wave front. In order to make the algorithm time-dependent, the speed of this wave front along a link is made to depend on when this wave front enters the link (e.g. Jacob *et al.*, 1999).

That is, for each link l we need a function $c_l(t)$ which returns the link "cost" (= link travel time) for a vehicle entering at time t. This information is calculated from the events taken from a run of the mobility simulation. In order to make the look-up of $c_l(t)$ reasonably fast, we aggregate over 15 min bins, during which the cost function is kept constant. That is, all vehicles entering a link between e.g. 9 AM and 9:15 AM will contribute to the average link travel time during that time period.

4.6 Time choice module (Time mutator)

In general, we want as a second module one that generates (or modifies) agents' activity schedules, which form the basis of their plans (Vaughn *et al.*, 1997; Bowman, 1998). One can divide

the task of creating an activity schedule into three parts: pattern choice, location choice, and timing choice. The *pattern choice* determines which activities to perform during the day, and in what order. For example, an agent might decide whether or not to go shopping, and if so, before or after work. The *location choice* determines where the agent will perform each activity. For some activities, such as home and work, agents make this decision very infrequently, while for other activities, such as shopping, the agent might choose a different location each time it wants to perform the activity. For example, an agent could go shopping at a bakery close to home or a grocery store near work. The *time choice* determines the duration of each activity, including when to leave home at the start of the day. A single module can make all of these decisions at the same time, or separate modules can be used to handle each one individually.

For this particular study, there is only a time choice module. This module takes the existing times of the plan and modifies them randomly. Note that there is no "goal" with this module, that is, the module does *not* try to improve any kind of score. Rather, the module makes a random modification, and the plans selection mechanism in conjunction with the scoring will make the agents improve toward better scores.

The exact details of the time mutator are as follows. This module reads the plans file, and for each plan alters the end time of the first activity by a random amount r_1 uniformly selected in the range $r_1 \in [-30min, 30min]$. Values that come before 00:00 are reset to that time. It then alters the duration of each activity except the first and last by separate random values uniformly selected from the same range. The last activity does not need modification since it runs from whenever the agent arrives until 24:00. The modified plans are written back out to a file.

4.7 Scoring

The utility function we currently use is described in detail in Charypar and Nagel (2003). For the convenience of the reader, we summarize it here, though it is sufficient to recognize that performing activities is rewarded, and travel, early arrival, and late arrival are punished. The utility function essentially translates the layout of the plan into a numerical value, which can be thought of as a score or an actual value in monetary terms.

Agents have a preferred duration for each activity, called the optimal duration, and represented by d_{opt}. The utility for performing an activity ($util_{perform}$) is a logarithmic function of the duration of time spent performing the activity. It is calibrated so that performing the activity for exactly d_{opt} hours causes the marginal utility to equal a fixed value, $\beta_{perform}$, which is the same for all activity types. In addition, it sets the utility for performing an activity for exactly d_{opt} hours to be $10\ h \cdot \beta_{perform}$. Mathematically,

$$util_{perform}(d) = \beta_{perform} * \left(10\ h + d_{opt} \log\left(\frac{d}{d_{opt}}\right)\right).$$

The result of the log function is that staying longer than d_{opt} always gains more utility, but each additional hour gains less utility than the preceding hour. Charypar and Nagel (2003) sets $\beta_{perform}$ to €+20/h, though for this study we set it to €+6/h. The reason for this change is explained below.

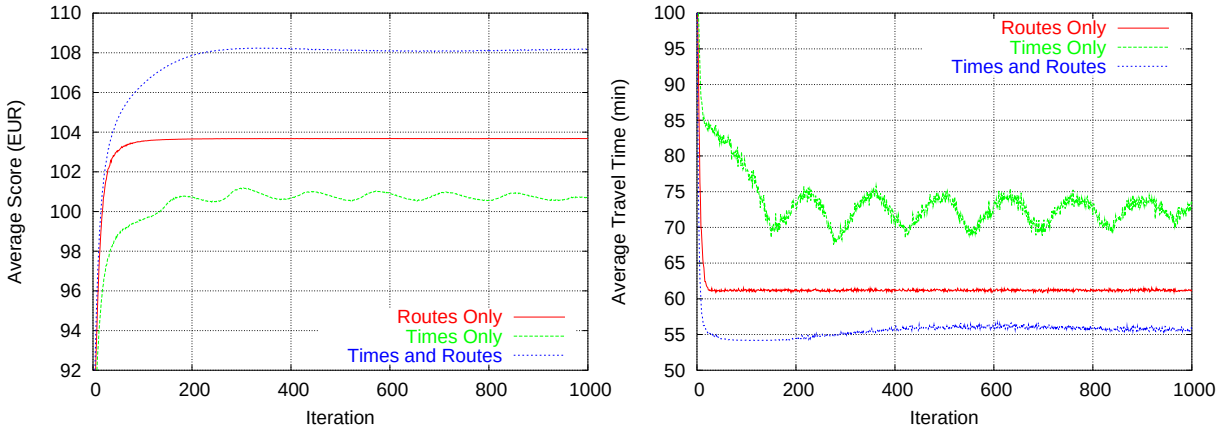
As mentioned previously, we have only two types of activities: home and work. The preferred duration of work time is 8 hours, and the preferred duration of staying home is 16 hours, so that the agents have no free time; i.e. their day is "full." In addition to duration, we set the useful hours for performing the work activity to between 7:00 AM and midnight, with the desired starting time to be exactly the opening time of 7:00 AM. The home activity has no time constraints. The utility of an agent who arrives to work before 7:00 AM must wait until 7:00 to start working. While waiting, the agent suffers a penalty defined by the marginal utility of waiting, β_{wait}, times the number of hours the agent waits, t_{wait}. Agents arriving after 7:00 AM are assessed a lateness penalty, determined by the marginal utility of being late, β_{late}, times the number of hours the agent is late, t_{late}. The final component of the utility is the utility of travel, which is the marginal utility of travel, β_{travel}, times the number of hours the agent spends traveling, t_{travel}. We set the above parameters to the following values: β_{wait} = €0/h; β_{travel} = €-6/h; β_{late} = €-18/h.

Given a full day plan, the agent prefers to spend all 24 hours in the day performing some activity. Any time spent by the agent not performing an activity causes a loss of potential utility. The agent would obtain a perfect score if it spent exactly 8 hours at work and 16 hours at home, allowing it to earn utility every minute of the day. This would earn the agent €60 for each activity, for a total score of €120. In an actual day, the agent must spend some of its time traveling. While traveling, the agent does not earn any utility for being at work or at home, and is simultaneously losing utility for being on the road. This means the agent incurs a double penalty: the actual negative utility for the travel itself, and the loss of potential positive utility for not performing an activity. Similarly, if the agent arrives early to work, it is also not spending time working (or being at home) so it loses potential utility. Note that being late has a real penalty, but does not cause any loss of potential utility, since the agent begins performing the activity immediately upon arrival.

Therefore, there is an indirect penalty associated with arriving early to a location (waiting) and traveling. If the agent's activity durations are near the optimum, this penalty is approx. $-\beta_{perform}$ times the number of hours spent waiting and/or traveling. Since agents will do their best to allocate time to the activities, we can assume that the durations are as near to optimum as possible. Given that the total travel time is on the order of 1 h, this is a reasonable assumption. Thus, the effective values of the above parameters are approximately: β_{wait} = €-6/h; β_{travel} = €-12/h; β_{late} = €-18/h. These effective values are selected such that, in rough terms, they model the Vickrey model of time choice (e.g. Arnott *et al.*, 1993). The desire to match the Vickrey model explains our choice of €+6/h for $\beta_{perform}$; if it was €+20/h, the effective penalty for arriving early would be of greater magnitude than the penalty for arriving late.

As mentioned earlier, the β value used by the agent database to multiply agents' scores affects their selection of those plans. This value can be considered to be a scaling function, which maps utility in Euro to unit-less values for the logit selection. In real-world applications, β will be estimated together with β_{wait}, β_{late}, and β_{travel}. This paper uses a value of $\beta = 2/€$ as a baseline value, and then looks at deviations from that value in Sec. 5.4. In estimated multinomial logit models, it seems that values between $1/€$ and $10/€$ are normal.

Figure 6: Relaxation of scores and travel times for all three scenarios of the baseline case. These plots display the average values of the (a) score and (b) travel time collected over the entire population of agents during each iteration.



(a) Average Scores

(b) Average Travel Times

5. Results

In this section we describe the results obtained from three scenarios based on the modules, parameters, network and initial conditions described above.

5.1 Routes Only

In the Routes Only scenario, we run the framework with the times replanning disabled, so that only route replanning may occur (i.e. $p_{times} = 0$). All agents are forced to forever use the initial activity time values, departing home at 6:00 AM and staying at work exactly 8 h. This scenario demonstrates how well the agents distribute themselves among the available routes.

Figure 6 shows the relaxation/learning behavior of the agents within this scenario, using two global performance measures: overall average score, and overall average travel time. One sees here that the average score relaxes to about €103.5 within 100–150 iterations, while average travel time relaxes to about 61 min within 20–30 iterations. Compared to the free-speed travel time of 54 min, the agents lose about 7 min due to congestion in this scenario.

Figure 7 shows the departure and arrival time distributions for this scenario, at iteration 0 (Fig. 7 (a)) and iteration 250 (Fig. 7 (b)). One can see from this figure that the work arrival time distribution (WATD) starts out at iteration 0 with an average of about 80–85 veh per 5 min time-bin, corre-

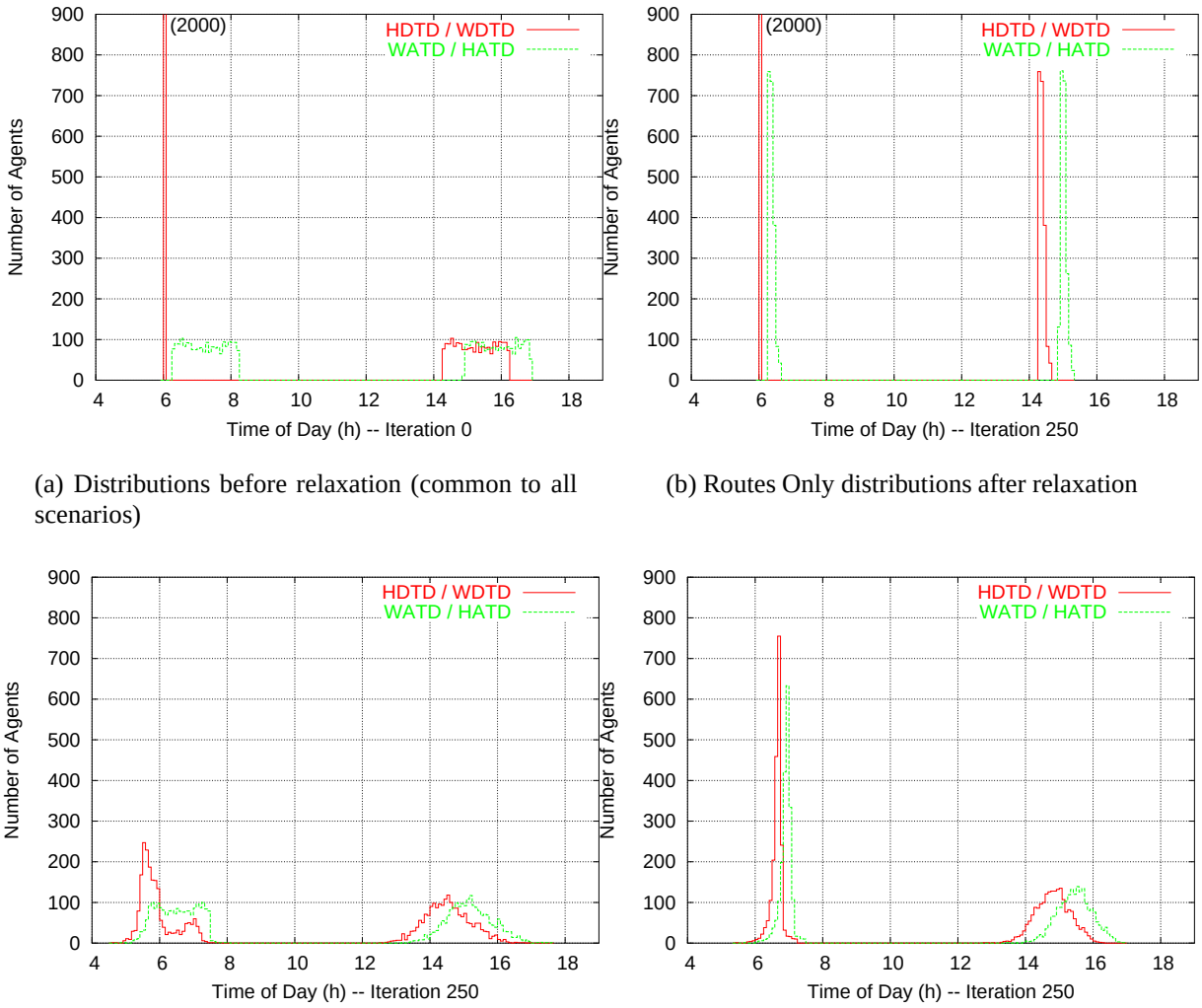
sponding to roughly 1'000 veh/h, and lasting for about 2 hours. This makes sense, as the capacity of the bottleneck for a single home-to-work route is 1'000 veh/h and there are 2'000 veh that want to traverse that route. After 250 iterations, the arrival rate increases to about 750 veh/5 min, or 9'000 veh/h, which corresponds to the total capacity of nine routes of 1'000 veh/h each. After 15 minutes, over 90% of the agents have arrived at work, with the remaining arriving in the next 10 minutes. This extra 10 min comes from the incomplete equilibrium in the route distribution caused by the 10% route replanning, which is explained further below. Since agents cannot change their work duration in this scenario, the work departure time distribution (WDTD) is the same as the WATD shifted by 8 hours, and since there are no bottlenecks on the route home, the home arrival time distribution (HATD) is the same as the WDTD shifted by 39 minutes.

The only degree of freedom in this scenario is the route choice. Figure 8 (a) displays the usage of the different routes as a function of iteration. This figure has several features. First, as expected, all agents start out using the initial route (number 7, "middle"), while the other eight routes (represented in the figure by route numbers 5 and 9) start out with no agents. Second, the percentage of agents using the middle route decreases at approximately a negative exponential rate. This makes sense, since 10% of all agents perform replanning each iteration. Some agents return to the middle route due to random plan selection, but most will stay on the other routes, lowering the percentage of agents using the middle route by roughly 10% of its previous value each iteration, until the agents are using all routes equally. It takes about 40 iterations for the middle route to have about the same usage percentage as the other routes.

The third feature of this figure is that after equilibrium most routes appear to be used on average by 10% of the agents at a time, rather than the $11.\bar{1}\%$ expected when nine equivalent routes are available. In addition, some routes appear to be used by 20% of the agents during certain iterations. These phenomena are explained by the fact that 10% of the agents perform replanning in each iteration, leaving the other 90% to choose freely which route they want to use. These 90% split up approximately evenly among the nine available routes, giving each a usage of about 10%, and the 10% who replan tend to choose the same route. This route will then have a total usage of about 20%. Only three routes are displayed here; if all nine were displayed, there would be a "spike" of 20% route usage occurring for some route in each iteration. This extra group of agents using one particular route causes that route to empty out later than the rest, extending the length of time agents arrive at work, as seen in Fig. 7 (b). Agents who replan tend to choose the same route because of the fluctuations in the usage of a route. During a given iteration, some route will happen to be used least, and thus have the best travel time, so the router will use it for all (or most) of the replanned routes in the next iteration. These fluctuations are also driven by the fact that slightly more or less than 10% of the agents may be replanned in each iteration, due to the probabilistic selection of agents for replanning. Note that it is not until near iteration 100 that the spikes appear to be in equilibrium.

The general interpretation of the above results for the Routes Only scenario is that the agents equilibrate to the different routes as best as they can, and once equilibrated stay in a very stable arrangement.

Figure 7: Histograms showing home departure time distribution (HDTD), work arrival time distribution (WATD), work departure time distribution (WDTD), and home arrival time distribution (HATD) of the three scenarios, before and after relaxation (250 iterations). The histograms are taken over 5 minute time-bins. For clarity the range stops at 900 vehicles, though some of the initial departure peaks are above this value; these peaks are labeled with their actual values



(a) Distributions before relaxation (common to all scenarios)

(b) Routes Only distributions after relaxation

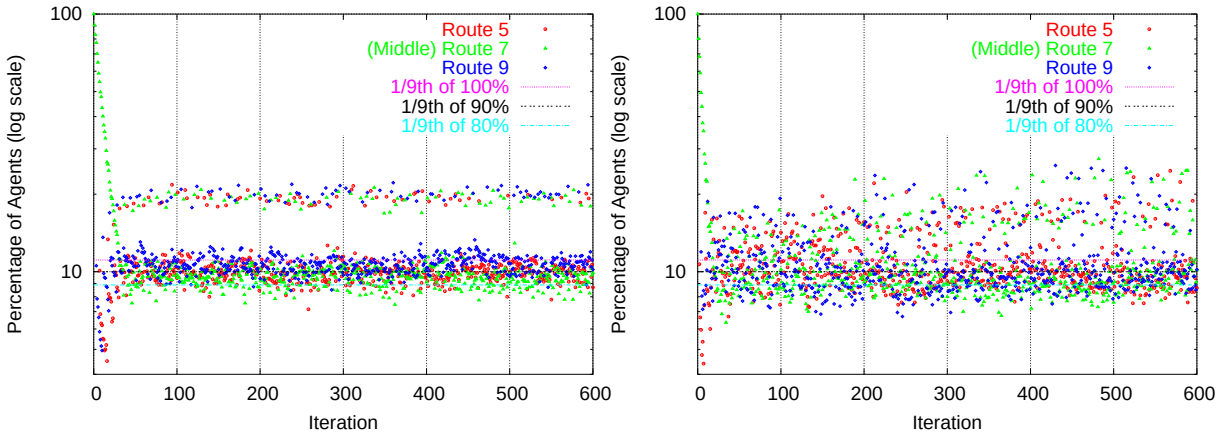
(c) Times Only distributions after relaxation

(d) Routes and Times distributions after relaxation

5.2 Times Only

In the second scenario, Times Only, we run the framework with the route replanning disabled, so that only times replanning may occur (i.e. $p_{routes} = 0$). Here all agents must forever use the middle route for their trips from home to work. This scenario demonstrates how well the agents

Figure 8: Route distributions for (a) the Routes Only scenario and (b) the Routes and Times scenario. The middle route has a distinctive curve since it is the one initially used by all agents. The other eight routes have qualitatively similar curves as Routes 5 and 9, only the "spikes" occur in different iterations.



(a) Routes Only

(b) Routes and Times

distribute themselves through time; i.e. how they handle peak-hour spreading.

Figure 6 includes the average scores and travel times for this scenario. One can see that these measures contain a considerable amount of oscillation in comparison to those of the other two scenarios, though the oscillation appears to diminish as the iterations continue. We presume that the oscillations are due to the system being in a chaotic regime, though more investigation is necessary to learn the exact cause. For now we only observe that they exist.

The average score oscillates around about €100.7, moving between €100.5 and €101.2, taking at least 200 iterations to reach this state. The average travel time centers around 72 min, oscillating between 68 min and 75 min, again taking about 200 iterations to get to that state. This scenario seems to find the worst scores and travel times of the three. It makes sense that the average travel times come out worse, since the agents cannot get around the 1'000 veh/h bottleneck of the middle route, while agents in the other scenarios can use nine times the capacity of this route for their home to work trips. This in turn explains why the average score is the lowest, because with more time being spent by the agents in travel, they have less time to spend working or at home, so they lose more potential score, which lowers their best achievable score.

Figure 7 shows the departure and arrival time distributions for this scenario, at iteration 0 (Fig. 7 (a)) and iteration 250 (Fig. 7 (c)). One can see that after 250 iterations, the WATD is still spread out to 2 hours and is still limited to 1'000 veh/h. This is as much as can be expected when all agents use the same route. The main peak of the home departure time distribution (HDTD) is shifted to about 5:30am, an earlier time compared to the 0th iteration, with a secondary peak around 7am.

Figure 9: Histograms showing HDTD and WATD for the Times Only scenario, after relaxation (250 iterations). The histograms are taken over 5 minute time-bins.

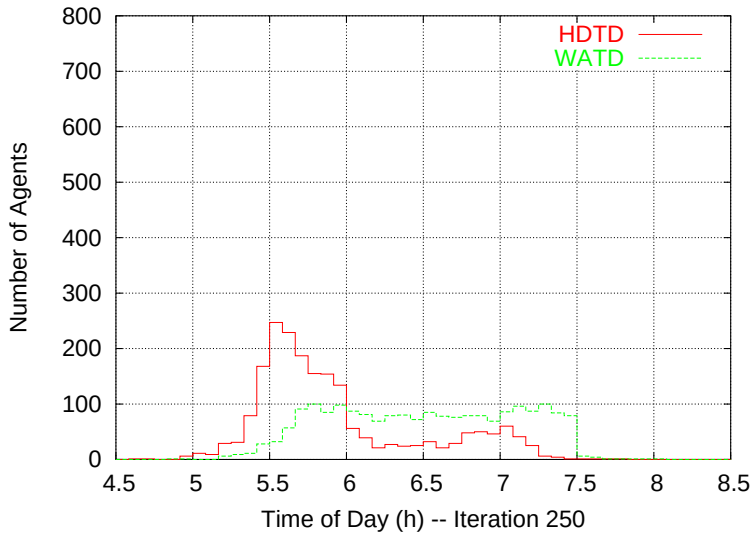


Figure 9 shows a close-up of the distributions during the morning rush-hour. In this figure one can see that about 3/4 of the agents arrive in the 90 min before 7am, and about 1/4 arrive in the half hour after. This makes sense, because agents arriving to work early are effectively penalized €-6/h while those arriving late are penalized at three times this amount. So, an agent arriving 30 min late incurs the same penalty as one arriving 90 min early. Back to Fig. 7 (c), we see that the WDTD and HATD are much more spread out than the WATD, lasting about 3 hours.

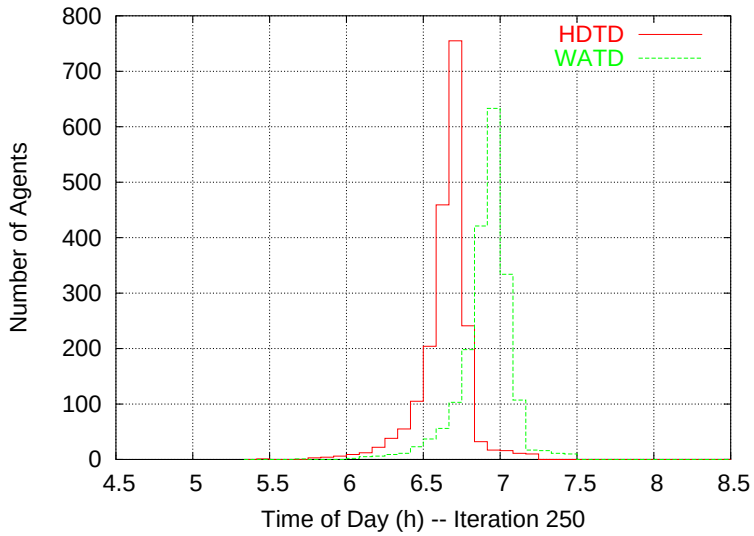
The general interpretation of the results for the Times Only scenario is that even with a time choice module that simply mutates existing plans, the feedback mechanism and the agent database allow agents to learn enough about the system to find a plausible distribution of departure times.

5.3 Routes and Times

In the Routes and Times scenario, we finally allow agents to utilize both the routing module and time choice module to develop new plans. Agents who perform time replanning also perform routes replanning on the resulting plan, as discussed in Sec. 4.4. This scenario demonstrates the complete relaxation behavior of the agents, where they may spread out over space and time.

Figure 6 includes the relaxation of scores and travel times for this scenario. One sees here that the average score is never perfectly relaxed, with what appears to be a slight oscillation with a period of about 800 iterations. However, after about 300 iterations the score seems to be rather stable, oscillating around €108. The average travel time initially finds the free-speed travel time within 100 iterations, then deviates from this value, eventually flattening out at about 55 min. The travel

Figure 10: Histograms showing HDTD and WATD for the Routes and Times scenario, after relaxation (250 iterations). The histograms are taken over 5 minute time-bins.



times may also have a oscillation, though it might also be a one-time "bump." More iterations would be required to find this out. It seems reasonable that this occurs because the agents are able to compensate for slightly varying travel times, meaning the travel time is not as important to them when they have more degrees of freedom to explore. In any case, this scenario finds a better average score and better average travel time than the other two scenarios, as expected given the larger number of degrees of freedom given to the agents.

Figure 7 shows the departure and arrival time distributions for this scenario, at iteration 0 (Fig. 7 (a)) and iteration 250 (Fig. 7 (d)). In iteration 250, the HDTD peak has shifted to about 6:45am, a later time than that at iteration 0, or that at iteration 250 for the other scenarios. It makes sense that the peak is at a later time than that of the Times Only scenario, as the average travel times in this scenario are shorter. The time of 6:45am makes sense as well, because most agents only need 15 min for the home to work trip. This is supported by the narrow WATD peak, which indicates that most agents arrive to work between 6:50am and 7am. The peak is nearly the same as the HDTD peak, only shifted by 15 min. See also Fig. 10 for a closeup of those peaks. Naturally, both the HDTD and WATD peaks are wider in this scenario than in Routes Only, since the agents can explore alternate departure times from home. They are not as wide as those in Times Only, since agents can also take alternate routes to avoid congestion, and don't have to spread out in time as much.

Figure 8 (b) displays the usage of the different routes as a function of iteration. As with the Routes Only scenario, all agents start out using the middle route, while representative routes numbers 5 and 9 start with no agents. Also like in the Routes Only scenario, the percentage of agents using the middle route decreases rapidly while percentage of agents using the alternate

route(s) increases. However, since 20% of the agents are given the chance to change their routes each iteration ($p_{times} + p_{routes}$), the exchange of agents from the middle route to the other routes occurs more rapidly.

This figure shows higher oscillations in route usage compared to the Routes Only scenario. In that scenario, route equilibration is the only option for agents trying to avoid congestion. Agents using some route tend to "notice" other agents using the same route, in the sense that their trip was made longer by the presence of the other agents. In this scenario, however, agents can also avoid congestion by choosing different departure times. So, agents using the same route may do so at totally different times, any may not notice each other at all, since they did not encounter any congestion from other agents along that route. Thus, they do not have much reason to try to switch routes, causing less of an equalization among the route choices. Another way to put it is that the temporal spreading allows the routes to remain equivalent to each other, even if the number of agents using each route differs greatly.

The general interpretation of the results for the Routes and Times scenario is that both modules work together well to allow the agents to explore both spatial and temporal degrees of freedom to obtain better plans than possible with just one degree of freedom.

5.4 Varying β

Here we vary the β plan selection parameter to higher and lower values from the baseline value of 2/€, to see how selecting the best plan more or less often affects the score and travel time relaxation rates. We tried these values for β: 0.01/€, 0.1/€, 1/€, 2/€, 4/€, 10/€, and ∞/€. A value of ∞/€ means agents *always* choose the plan with the best score.

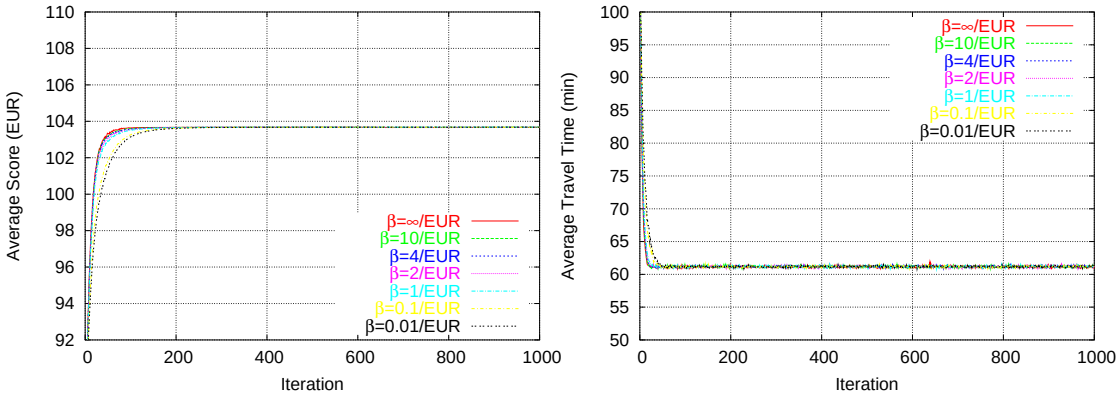
Figures 11 (a) and (b) show the effect of β on the Routes Only scenario. The relaxed score and travel time averages are the same; only the rate of approach to those values differs. With a lower value of β, agents are allowed more random selection among their plans, so the system approaches the steady-state at a slower rate, which makes sense. The infinite β, which causes agents to always choose the best plan they have, allows for the fastest relaxation of both scores and travel times.

Figures 11 (c) and (d) show the effect of β on the Times Only scenario. One can see that the oscillations have a higher amplitude and lower frequency for lower values of β. For the scores, all the curves seem to have roughly the same worst score (lower bound) of about 100.5. However, with lower β the system is able to find better (higher) scores, though it cannot stay at those values. Similarly, on the travel time plots, one can see that the worse possible travel time (upper bound) doesn't change much, but better (lower) travel times are reached with lower values of β.

Figures 11 (e) and (f) show the effect of β on the Routes and Times scenario. Like with the Routes Only scenario, the relaxed value of the score seems to remain essentially the same, but the different β values approach it differently.

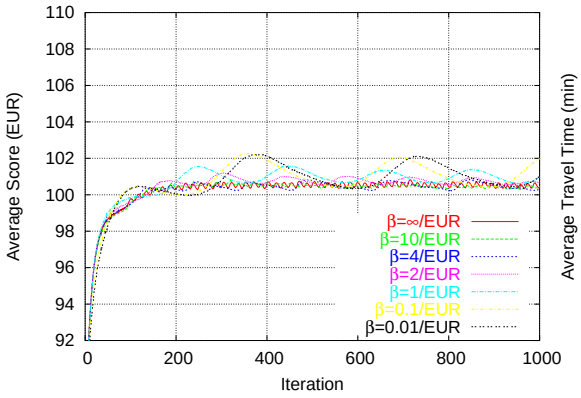
Overall, it appears that value of β does not matter very much for the scenarios with the routing

Figure 11: Relaxation of scores and travel times for the Routes Only scenario, comparing the relaxation behavior with varying β values. These plots display the average values of the score (left) and travel time (right) collected over the entire population of agents during each iteration.

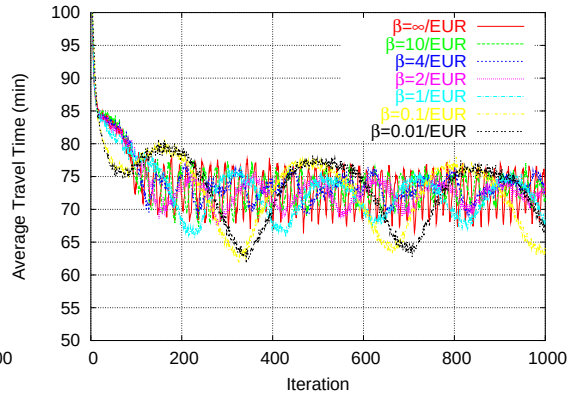


(a) Average Scores for Routes Only

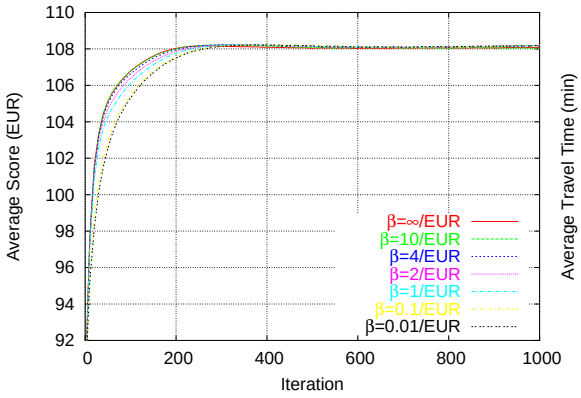
(b) Average Travel Times for Routes Only



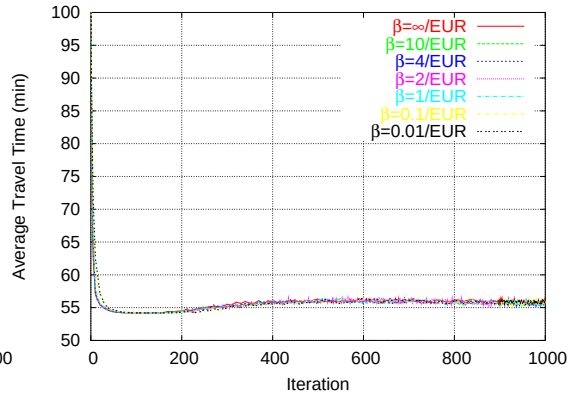
(c) Average Scores for Times Only



(d) Average Travel Times for Times Only



(e) Average Scores for Routes and Times



(f) Average Travel Times for Routes and Times

module enabled. Perhaps this is due to the fact that the routing module make decisions with some "intelligence" behind them, allowing for an additional learning mechanism for the agents. Possibly, the Times Only scenario is affected more by the value of beta, as the decisions made by the agent database are the only ones that have any effect on the learning behavior.

5.5 Varying β_{travel}

Here we vary the marginal utility of travel time, β_{travel} from its baseline value of €-6/h, to see how making travel time more or less important in the score calculation affects the score and travel time relaxation rates. We tried these values β_{travel}: €-0.06/h, €-0.6/h, €-6/h, €-60/h, and €-600/h.

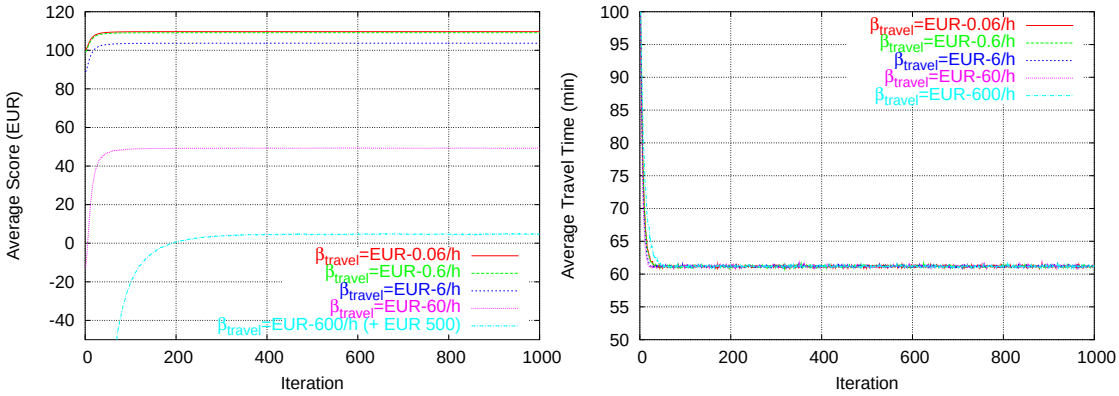
Figures 12 (a) and (b) show the effect of β_{travel} on the Routes Only scenario. As expected, higher magnitudes of β_{travel} cause the average score to relaxed to a lower value, since all else being equal, the same travel time costs more to the agent. For all values above €-600/h, the curves seem to have the same relaxation behavior as seen in Fig. 6. For β_{travel} =€-600/h, the score takes about 200 more iterations to relax, while the travel time takes only 10–20 more iterations to relax.

Figures 12 (c) and (d) show the effect of β_{travel} on the Times Only scenario. Here we see that different values of β_{travel} can also change the oscillation amplitude and frequency for the scores and the travel times. For β_{travel} =€-60/h, the oscillation frequency is higher, and the amplitude is smaller, and for β_{travel} =€-600/h, the oscillation is nearly nonexistent. For the smaller two magnitudes of the marginal utility of travel, the score and travel times curves look qualitatively like those of the Routes and Times scenario in Fig. 6. They improve at first, then slightly deviate from the best value obtained. This supports the idea that the behavior of the Routes and Times scenario comes from the fact that travel time is less important to the agents when they are able to adjust their routes and their activity schedules simultaneously. In addition, as with the Routes Only scenario, Times Only takes longer to relax when the β_{travel} value is higher.

Figures 12 (e) and (f) show the effect of β_{travel} on the Routes and Times scenario. Here we once again get basically the same relaxation behavior, offset only by the different strengths of the travel time in the overall score. The scores for β_{travel}=€-600/h takes longer to relax, but all scores for the Routes and Times scenario relax to higher values than those of the other two scenarios. Furthermore, one can see that for the lower marginal utility of travel, the travel times stay flat at close to the free-speed travel time. The deviation from this level occurs more for higher values of β_{travel}, with β_{travel}=€-6/h being the only one that deviates and appears to return to the lower value.

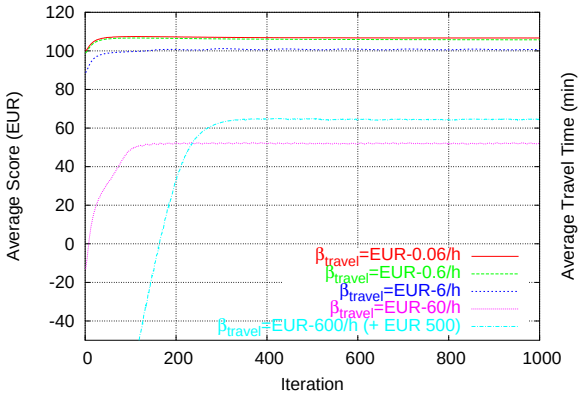
Overall, it appears that reasonable values of β_{travel}, as compared to the other marginal utility parameters, lead to the same relaxation behavior (if not the same scores) in the scenarios with the routing module enabled. The Times Only scenario's relaxation behavior is more affected by the value of β_{travel}, and it appears that a value of €-6/h may represent an unstable case at the border between two more stable regimes: one where travel time dominates the decision making, and one

Figure 12: Relaxation of scores and travel times for the different scenarios, comparing the relaxation behavior with varying β_{travel} values. These plots display the average values of the score (left) and travel time (right) collected over the entire population of agents during each iteration. For better comparison of the relaxation rate, we shift the average score curve for β_{travel} =€-600/h up by €500.

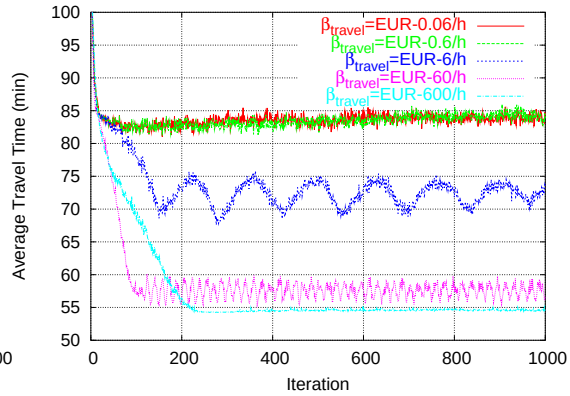


(a) Average Scores for Routes Only

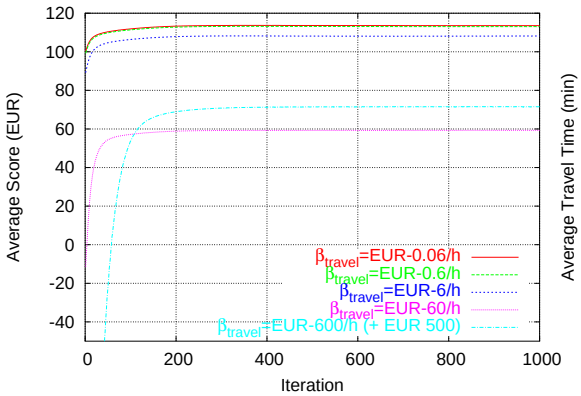
(b) Average Travel Times for Routes Only



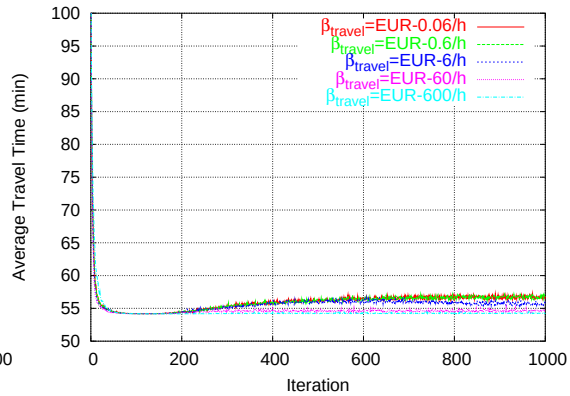
(c) Average Scores for Times Only



(d) Average Travel Times for Times Only



(e) Average Scores for Routes and Times



(f) Average Travel Times for Routes and Times

where it is not very important at all.

6. Discussion and future Work

We have shown that the system can relax to an uncongested state using the present times mutator module, however the relaxation takes many hundreds of iterations. We expect that a more goal-oriented module, which tries to return new activity time schedules that are better than old ones, would allow the system to relax faster. We are working on two versions of such module at different stages of development and capability. Both of them use genetic algorithms (GA) to "evolve" activity schedules by mutating or mixing existing schedules. One of these contains a global (for all agents) mental map of the traffic network for estimating travel times between proposed activities, but only adjusts the durations of the activities, leaving patterns and locations alone. Unfortunately this module has some bugs in the mental map that have not been worked out yet, so it generates faulty schedules (Schneider, 2003; Raney *et al.*, 2003). The second GA-based model has no known bugs but is not yet integrated into the framework (Charypar and Nagel, 2003).

In addition, we plan to add a population generation module that would disaggregate demographic data to obtain individual households and individual household members (agents), with certain characteristics, such as a street address, car ownership, or household income (Beckman *et al.*, 1996).

As mentioned in Sec. 3.5, the agent database must be aware of dependencies between modules. At this stage, with only two modules, it is easy to hard-code this dependency. However, once we begin adding more modules, we will need to make this information configurable in some way. In fact, it might be desirable to have different "module paths" an agent can carry out, with different execution probabilities. For example, an agent could be given the choice between calling only the activity time choice module, calling only the routing module, or calling both (perhaps even with different choices for the calling order). The combination and probabilities of the available choices could reflect behavioral aspects, such as the low frequency of changing work locations (with a work location choice module) compared to the high frequency of changing shopping locations or durations.

Currently a lot of time is spent writing and reading files to communicate plans between the agent database and the modules/simulation. We are working on including a message-based technology into the framework so that plans can be sent directly between the modules as needed, and plans and events can be sent to and from the simulation while it is running. This change will allow decision-making and simulation to occur simultaneously, meaning this change would allow us to implement within-day replanning.

As mentioned in the previous section, we intend to eliminate the memory size limitation imposed on the agent database by current 32-bit architecture by spreading it over many CPUs. The agent database on each CPU would maintain the plans and scores for a subset of the agents in the simulation and would utilize its own set of modules to update those plans. The different plan-sets

would be merged when sent to the simulation.

7. Summary / Conclusions

This paper presents a framework for large scale multi-agent simulations of travel behavior. The immediate goal is a replacement of the traditional four-step process for transportation planning; the longer term goal is to have an agent-based system for all aspects of urban and regional planning. The core ideas of the system are:

- One should separate the simulation of the physical world from the simulation of the mental world of the agents. This is reflected by a design that strictly separates those two worlds. A mental/strategic layer generates strategies ("plans"); these plans are executed by a physical layer; and then strategies are adapted based on how they performed in the physical layer. In the specific case of transportation simulations, the physical layer is also called the traffic (micro-)simulation, or the mobility simulation.

- Agents should memorize more than one strategy. This gives them the option to try out all strategies multiple times, and thus to obtain average scores for them ("exploration"). When agents do not explore, then they exploit, by selecting strategies based on the score, possibly using some behaviorally justified model such as multinomial logit. From time to time, agents should also add new strategies to their repertoire, or delete strategies with low scores. All this functionality is implemented by a so-called agent database.

- As said above, agents from time to time add new strategies to their repertoire. These strategies need to be generated by some method. In our approach, the methods that generate new strategies can be completely independent modules. They read an XML file that specifies which pieces of an agent's plan are already known, and the external module adds or changes pieces. For example, one module could generate activity patterns, another one activity locations, a third one activity times, and a fourth one routes. The framework will call all those modules one by one and in the end have a completely new plan. The XML plans file uses exactly the same format for all those exchanges; and the same format is again used to send plans to the physical simulation. This means that we have only one file format specifying agent plans.

- Performance information from the physical simulation is communicated back via an events file. Events are output every time something relevant happens, such as an agent leaving an activity, or entering a link. In general, the physical simulation performs no data aggregation at all; in consequence, something like link travel time needs to be constructed from link entry and link exit events. The two big advantages of this approach are: (1) It is straightforward to retrofit those outputs to any existing agent-based mobility simulation, since there is no data aggregation either along space or along time. Any event is just a simple print command inside the mobility simulation. (2) External strategy generation modules can also read this information and use it as they like. For example, an external module building

a mental map for one specific agent will use the link entry/exit information in completely different ways than the routing module. If the simulation would aggregate link entry/exit information into link travel times, adding such a mental map module would no longer be possible.

- Strategies are full 24-hour day plans, and are also evaluated as such. The strategy evaluation is based on the events, and therefore on actual individual performance of each agent. The evaluation is therefore immune against errors introduced by aggregation. One obvious candidate for the scoring function is the conventional utility function, but other systems such as aspect theory can be used. Since all agents are individually represented, it is no problem to couple those evaluations to individual attributes, such as gender or income.

- The approach is completely transparent toward parallel computing. Having a parallel mobility simulation needs some programming effort, but is conceptually straightforward since methods from the simulations of other physical systems can be applied. And making the agent database or the external strategy generation modules parallel is completely trivial as long as agents do not interact: One simply distributes agents (or possibly households) across CPUs. Only when interactions beyond the household level become important will more advanced computing techniques become necessary.

- The implementation was tested with three scenarios, one testing route equilibration, one testing times choice, and and one testing route equilibration in conjunction with time choice. Route equilibration works as expected, in the sense that the system ends up with using nine equivalent routes equivalently. Also time choice works as expected, in the sense that agents adjust their daily timings in a plausible way.

- A curious aspect about the time choice test is that the external module that generates alternative timing schemes has no “intelligence” at all: Instead, it just randomly mutates existing schedules of the agents. The interpretation of the new schedules is entirely done by the agent database and its scoring function. This means in practice that the correctness specifications toward external modules are significantly relaxed: In essence, it is sufficient if external modules generate meaningful solutions plus some variability.

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 mobility simulation. The work also benefited from discussions with Fabrice Marchal. Michael Balmer helped run the simulations from which the results presented here were obtained.

References

Arentze, T., F. Hofman, H. van Mourik and H. Timmermans (2000) ALBATROSS: A multi-agent rule-based model of activity pattern decisions, *Paper*, **22**, Transportation Research Board Annual Meeting, Washington, D.C.

Arnott, R., A. D. Palma and R. Lindsey (1993) A structural model of peak-period congestion: A traffic bottleneck with elastic demand, *The American Economic Review*, **83** (1) 161.

Astarita, V., K. Er-Rafia, M. Florian, M. Mahut and S. Velan (2001) A comparison of three methods for dynamic network loading, *Transportation Research Record*, **1771** 179–190.

Avineri, E. and J. Prashker (2003) Sensitivity to uncertainty: Need for paradigm shift, *Paper*, **03-3744**, Transportation Research Board Annual Meeting, Washington, D.C.

Beckman, R. J., K. A. Baggerly and M. D. McKay (1996) Creating synthetic base-line populations, *Transportion Research Part A – Policy and Practice*, **30** (6) 415–429.

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

Bottom, J. (2000) Consistent anticipatory route guidance, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Bowman, J., M. Bradley, Y. Shiftan, T. Lawton and M. Ben-Akiva (1999) Demonstration of an activity-based model for Portland, vol. 3, Elsevier, Oxford.

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

Cetin, N. and K. Nagel (2003) A large-scale agent-based traffic microsimulation based on queue model, in *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, CH. Earlier version, with inferior performance values: Transportation Research Board Annual Meeting 2003 paper number 03-4272.

Charypar, D. and K. Nagel (2003) Generating complete all-day activity plans with genetic algorithms, in *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, August 2003. See http://www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.

Daganzo, C. (1998) Queue spillovers in transportation networks with a route choice, *Transportation Science*, **32** (1) 3–11.

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, *Networks and Spatial Economics*, **2(4)** 347–369.

DYNAMIT-www (accessed 2003) DYNAMIT, See its.mit.edu and dynamictrafficassignment.org.

DYNASMART-www (accessed 2003) DYNASMART, See www.dynasmart.com and dynamic-trafficassignment.org.

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

Friedrich, M., I. Hofsäß, K. Nökel and P. Vortisch (2000) A dynamic traffic assignment method for planning and telematic applications, in *Proceedings of Seminar K*, European Transport Conference, Cambridge, GB.

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

Gawron, C. (1998b) Simulation-based traffic assignment, Ph.D. thesis, University of Cologne, Cologne, Germany. Available via www.zaik.uni-koeln.de/˜paper.

Hensher, D. and J. King (2001) in D. Hensher and J. King (Eds.), *The Leading Edge of Travel Behavior Research*, Pergamon, Oxford.

Holland, J. (1992) *Adaptation in Natural and Artificial Systems*, Bradford Books. Reprint edition.

Hunt, J., R. Johnston, J. Abraham, C. Rodier, G. Garry, S. Putman and T. de la Barra (2001) Comparisons from sacramento model test bed, *Transportation Research Record*, (1780) 53–63.

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

Jonnalagadda, J., N. Freedman, W. Davidson and J. Hunt (2001) Development of microsimulation activity-based model for San Francisco: destination and mode choice models, *Transportation Research Record*, (1777) 25–35.

Kaufman, D. E., K. E. Wunderlich and R. L. Smith (1991) An iterative routing/assignment method for anticipatory real-time route guidance, *Tech. Rep.*, **IVHS Technical Report 91-02**, University of Michigan Department of Industrial and Operations Engineering, Ann Arbor MI 48109, May 1991.

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

Loudon, W., J. Parameswaran and B. Gardner (1997) Incorporating feedback in travel forecasting, *Transportation Research Record*, (1607) 185–195.

Nagel, K. (1994/95) High-speed microsimulations of traffic flow, Ph.D. thesis, University of Cologne. See www.inf.ethz.ch/˜nagel/papers.

Nagel, K. (1996) Individual adaption in a path-based simulation of the freeway network of Northrhine-Westfalia, *International Journal of Modern Physics C*, **7** (6) 883.

Nagel, K. and F. Marchal (2003) Computational methods for multi-agent simulations of travel behavior, in *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, August 2003. See http://www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.

Raney, B., M. Balmer, K. Axhausen and K. Nagel (2003) Agent-based activities planning for an iterative traffic simulation of Switzerland, in *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, August 2003. See http://www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.

Raney, B. and K. Nagel (in press) Iterative route planning for large-scale modular transportation simulations, *Future Generation Computer Systems*. See sim.inf.ethz.ch/papers.

Salvini, P. and E. Miller (2003) ILUTE: An operational prototype of a comprehensive microsimulation model of urban systems, in *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, August 2003. See http://www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.

Schneider, A. (2003) Genetische Algorithmen zur Optimierung von Tagesplänen für Verkehrsteilnehmer, Term project, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. See www.sim.inf.ethz.ch/papers.

TRANSIMS www page (accessed 2003) TRansportation ANalysis and SIMulation System, transims.tsasa.lanl.gov. Los Alamos National Laboratory, Los Alamos, NM.

URBANSIM-www (accessed 2003) URBANSIM, See www.urbansim.org.

Vaughn, K., P. Speckman and E. Pas (1997) Generating household activity-travel patterns (HATPs) for synthetic populations.

Vovsha, P., E. Petersen and R. Donnelly (2002) Microsimulation in travel demand modeling: lessons learned from the New York best practice model, *Transportation Research Record*, (1805) 68–77.

Wahle, J., A. Bazzan, F. Klügl and M. Schreckenberg (2002) The impact of real-time information in a two-route scenario using agent-based simulation, *Transportation Research C*, **10** (5–6) 399–417.