Truly Agent-Based Strategy Selection for Transportation Simulations

Bryan Raney
Dept. of Computer Science, ETH Zürich, CH-8092 Zürich, Switzerland
Telephone: +41 1 632 0892
Fax: +41 1 632 1374
E-mail: raney@inf.ethz.ch

and

Kai Nagel
Dept. of Computer Science, ETH Zürich, CH-8092 Zürich, Switzerland
Telephone: +41 1 632 2754
Fax: +41 1 632 1374
E-mail: nagel@inf.ethz.ch

Submission date: Nov 15, 2002

Corresponding author: Kai Nagel

Number of words: 7500

Abstract

Multi-agent transportation simulations represent travelers as individual "agents," who make independent decisions about their actions. We are implementing such a simulation for all of Switzerland, which is composed of modules that model those decisions for each agent, such as: (i) **Route planner:** Generates routes. (ii) **Micro-simulation:** Executes routes simultaneously; computes agent interactions, leading to congestion. (iii) **Feedback:** Iterates the above modules, resolving interdependencies. We discuss the operation of these modules, and focus on improvements made to the feedback system, such as an agent "memory" that allows agents to choose among previously used routes based on their past performance.

1 INTRODUCTION

During the last decades, simulation has joined theory and experiment as a third method of scientific inquiry. In simulation a virtual version of the real world is reconstructed inside the computer, and scientific and engineering questions can then be addressed to that computational model. Simulation is a particularly appropriate method when analytical theory alone has little predictive power, when controlled experiments are costly or impossible, or when extracting measurement data from the real system is costly or impossible. All three criteria are fulfilled in transportation science as well as in socio-economic science in general.

In addition, a new method is particularly appropriate for the simulation of socio-economic systems: the method of multi-agent or agent-based simulations. In these methods, all entities of the system are represented as objects which have internal states and follow individual rules. This is particularly appropriate for objects which have a complicated internal structure and which all differ from each other. Again, this is true for transportation science as well as in socio-economic science in general, but it also encompasses engineering objects such as adaptive traffic signals.

The emergence of the agent-based technique would not have been possible without the emergence of modern computing. Rule-based code runs efficiently on modern PCs, making the technology widely available. In addition, agent-based code runs efficiently on modern parallel supercomputers, while it did not run well on more traditional vectorizing supercomputers such as Cray. This makes the jump in computational capabilities even larger than, say, in computational fluid-dynamics. In consequence, it is now possible to perform agent-based simulations of large metropolitan areas with 10^7 or more travelers.

The last decades have seen considerable progress in making agent-based simulation technology applicable for transportation. Agent-based transportation simulation packages consist of many modules, including the traffic simulation itself, route generation, activity generation, housing choice, land use evolution, freight traffic, but also traffic management strategies. Most if not all groups concentrate on a subset of these modules.

The modules interact, and the interaction goes in both directions: for example, the execution of plans leads to congestion, yet the expectation of congestion influences plans. Any large scale transportation package needs to resolve this logical deadlock in a meaningful way. Reality seems to approach the issue of feedback by a slow system-wide learning process: People pre-plan major pieces of their life (like when and where they work) a long time in advance and normally only re-adjust small pieces of their schedules when needed (*1*). More precisely, they pre-plan and re-adjust on many time scales, where the time scale is related to the magnitude of the adjustment: workplaces and home locations are re-adjusted on time scales of several years, while the decision to make a detour to buy some ice cream may happen within seconds. In consequence, a simulation system is faced with two challenges:

1. Modeling adaptation and learning on all time scales — In principle, a transportation simulation should simulate several thousand days in sequence, and the decisions of the individual people should unfold on their particular time scales as pointed out above. In particular, travelers should be able to replan while en route. While this sounds simple in principle, it is difficult in practice, because one wants to avoid a large monolithic software package and thus one wants to separate the traffic flow simulation from the strategic decision-making of the travelers. This becomes particularly relevant for parallel transportation simulations, since now the strategic planning needs to be separated from the traffic simulation also for performance reasons. This is not the topic of this paper; see (*2*) for more information.

2. Behavioral realism vs. fast relaxation — In practice, simulating several thousand days in sequence is difficult to do because of computational resource limitations. It is also questionable if this would yield useful results without a deep understanding of the learning dynamics. As a reaction to this, mathematical modeling of transportation scenarios, as well as of economics in general, in the past has relied on the notion of a Nash or User Equilibrium (UE). As is well known, in a UE no traveler can improve by unilaterally changing her/his behavior. The advantage is that a UE prescribes a state of the system and it does not matter how the computational system arrives at it — as opposed to a realistic modeling of the transient learning dynamics. Today, we however increasingly recognize that socio-economic systems do not operate at a User Equilibrium point; for example, for the housing market it is assumed that the system is permanently in the transients (*3*).

This second point is the focus of this paper. Our approach to the problem is to design a framework which admits all the different views to the problem. That is, the framework should as well converge to the User Equilibrium (assuming it is unique and an attractor — this is a difficult discussion but again outside the scope of this paper) as it should allow for experimentation with different behavioral hypotheses. We entirely concentrate on day-to-day replanning although our results will also apply to within-day replanning. In particular, we will demonstrate that the introduction of an agent database, which keeps track of agents' past strategies and their performances, will greatly improve both plausibility and robustness of the system.

Throughout this paper we use the term *agent* to refer to an entity within the simulation capable of making decision about its actions (such as the route to take from point a to point b). Since our simulation does not yet involve land use or other non-transportation activities, an agent is presently equivalent to a *traveler*, a person using the transportation network.

The structure of this paper is as follows: Section 2 gives a general introduction to multi-agent simulations. This is followed (Sec. 3) by an overview of existing methods to solve the dynamic traffic assignment problem, which is the example problem which was used in our investigations. Section 4 describes the specific modules we are using in this study. Section 5 introduces the traffic scenarios we are applying those modules to. Following that, Sec. 6 describes some results from the day-to-day replanning of our feedback system, which turned out to have some implausible implications. We continue the section by describing some alterations we made to the feedback mechanism to try to resolve the problems, and the results of those changes. Next we present in Sec. 7 the agent database, a completely different and more robust approach to solving the problems encountered in the previous section. This section includes results from the agent database approach. We finish with a discussion and a summary.

2 MULTI-AGENT SIMULATIONS (MAS)

Arguably, one can differentiate between five broad categories of simulation techniques:

(i) Particle simulations (e.g. molecular dynamics)

(ii) Discretized field equations (partial differential equations)

(iii) Cellular automata

(iv) Discrete-event simulations (queueing models)

(v) Multi-agent simulations (MAS)

There are situations where particles are considerably more complex and possess considerably more internal "intelligence" than in standard physics or engineering applications. Typical examples of such particles are humans, but one can also look at ants or

adaptive traffic signals. Such particles can not be described by the methods (i)–(iv). A viable method is however to use rules to describe such particles. Rules can be easily encoded for computers, but they are difficult to treat using the methods of continuous mathematics. For that reason, MAS is a recent method.

MAS can be seen as a very recent merger of techniques from two different disciplines: (i) Complex Adaptive Systems (CAS), and (ii) Distributed Artificial Intelligence (DAI). Influences to CAS are cellular automata (*4*), neural networks (*5*), classifier systems/genetic algorithms (*6*), and ant colony optimization (*7*). interaction of many simple entities to produce complex behavior; in that sense, the CAS interpretation of a neural network would be that many simple neurons interact in order to produce complex emergent behavior of the neural network. One direction of CAS research has taken those complex entities composed of simple rules to again interact and thus form higher-level emergent complex adaptive systems. An example for this are stock market simulations using a different genetic algorithm for the simulation of each individual trading agent (*8*).

In contrast, Distributed Artificial Intelligence (DAI) has concentrated on the internal structure of individual agents. Recently, the interactions of several such agents has received much more attention (*9, 10*). Somewhat simplifying, one can say that DAI concentrates on the interaction of few relatively complicated agents, while CAS concentrates on the interaction of many relatively simple agents. Therefore, MAS, as a combination of CAS and DAI, concentrates on the interaction of many, somewhat complicated agents.

MAS codes are best implemented using object-oriented programming languages, since those languages allow the clean encapsulation of internal object structure and object-object interaction. An early project in this direction is SWARM (*11*) based on ObjectiveC; a recent addition is RePast (*12*) based on Java. People interested in computational performance will also consider C++.

With respect to transportation simulations, aspects of MAS are slowly entering the field. Let us look at transportation planning. The traditional approach is the four-step process, consisting of Trip Generation, Trip Distribution, Mode Choice, and Route Assignment (e.g. (*13, 14*)). When looking at each of the individual modules, one recognizes that they are supposed to describe human behavior, but they are formulated in terms of aggregated flows. For example, many mathematical formulations of the route assignment problem allow flows to be continuous variables (*14*).

There are however considerable efforts to put transportation simulation on a better behavioral basis (see (*15*)). These efforts include for example activity-based demand generation (e.g. (*16, 17*)) and individual route-choice models (e.g. (*18*)). In addition, traffic simulations which allow one to follow individual travelers (e.g. (*19, 20, 21, 22, 23*)) make it now possible to couple those behavioral demand generation models with plausible traffic dynamics, allowing for logically consistent feedback. For example, it was difficult to consider departure time choice and the resulting activity scheduling problem with static route assignment models which cannot model peak spreading. High-performance implementations of those simulations (e.g. (*24, 25, 26*)) allow the simulation of very large scale scenarios, with 10 million or more travelers simultaneously in the system.

Looking further ahead, the agent-based representation is extremely amenable to methodological developments. For example, results from psychology (e.g. (*27*)), from experimental economics (e.g. (*28*), (*29*)), or from travel behavior research (e.g. (*1*)) can be directly used for such simulations, without the need for further transformations into the variables used by the simulation system. In fact, we are currently in the process of implementing activity-based demand generation within the agent-based framework.

3 DYNAMIC TRAFFIC ASSIGNMENT (DTA)

The problem considered in this paper is known as dynamic traffic assignment (DTA). Given a set of travelers, all with fixed starting time, the task is to find plausible routes for each traveler. In order to define "plausible", one often makes the assumption that travelers should go towards a User Equilibrium (UE), where no traveler can arrive earlier at his/her destination by changing routes. However, the UE formulation is a mathematical construct; it is not the real problem that we want to solve.

The traditional approach to this type of problem is static assignment (e.g. (*14*)). In static assignment, there are is no time-of-day; instead, trips between any given origin-destination pair are assumed to occur at a constant rate. The advantage of the static formulation is that a fair amount of mathematics is known about it, including a uniqueness proof for certain situations, which allows one to compare results even if they are obtained with different methods.

Since the static assignment problem is non-linear, the methods to solve it are iterative (e.g. (*14*)). One of the oldest solution algorithms is the Frank-Wolfe algorithm. An interpretation of it is that the routes of a certain fraction of the population are re-planned optimally based on the current link travel times. Those routes then replace the previous routes of that part of the population. New link travel times are computed using the new set of routes and the volume-based link travel time function. A critical issue is the selection of the replanning fraction; it turns out that the Method of Successive Averages (MSA), which means that the replanning fraction drops as $1/n$ where n is the iteration number, guarantees convergence, albeit a slow one, to the correct (unique) solution.

Static assignment does not, by definition, model time-dependent dynamics correctly. There are many ways to attack this problem, ranging from making relatively small changes to static assignment (e.g. (*30*, *31*, *32*)), via so-called mesoscopic simulations (*33*, *20*, *21*, *34*), to very detailed micro-simulations (*19*, *22*, *35*, *36*).

Since much less is known about the mathematics of the dynamic problem (*37*), implementations often move to "learning algorithms" (*13, 30*) instead of the nonlinear optimization methods used for the static assignment. In learning algorithms, the travelers search for better routes based on some kind of historic information; there are many different ways of doing this. The learning algorithms typically no longer guarantee any property of the solution (such as being a User Equilibrium), but in general one can assume that they go towards a fixed point (if everything is deterministic) or towards a steady state density (for stochastic systems if they are Markovian) (see, e.g., (*38*)). Markovian means that the transition probabilities to the next iteration only depend on the current iteration. If the stochastic systems are in addition ergodic, that is, each possible combination of routes can be reached by the iterative dynamics, then the steady state density is unique (*39*). For certain non-traffic systems it has even been shown that a best-reply dynamics leads to a User Equilibrium, an observation which connects learning dynamics and optimization techniques.

Unfortunately, for large scale scenarios these properties are less useful than they sound. Let us concentrate on stochastic systems. The problem essentially is that the relatively small number of iterations that is typically done is by far not enough to come close to the mathematical limit. Intuitively, with a plausible 10% replanning fraction, ten iterations are necessary until everybody in the population had one chance to react to a change in the system. However, every time someone finds a solution (a route) that was not previously used, this counts as such a change and the ten "final" iterations need to start over. A drastic version of this is called "broken ergodicity" (*40*), where the system can remain in subregions of the phase space for very long times, in spite of being ergodic.

In practice, however, such problems with dynamic traffic assignment rarely occur and simulations relax reasonably fast (*41*, *38*). However, none of the currently known mathematics explains this nice behavior; maybe one can speculate that the dynamic problem inherits some of the "niceness" of the static problem.

Existing learning approaches to the dynamic traffic assignment typically only remember one route per traveler. Some approaches (e.g. (*20*)) use a probabilistic route choice which is somewhat similar to our agent-based approach, but in general the thinking behind those methods is different: In those implementations, the assumption is that there is some external module which calculates all options for each traveler, and then makes a random weighted draw between these options (discrete choice theory, see e.g. (*16*)). The external module needs to compute the utilities for each individual option. Since it is not possible that all options have been previously tried out, the external module needs to make assumptions about the option's performance. This enters a possible cause of inconsistencies into the system, since those assumptions about system performance need not be identical with actual performance. Such inconsistencies can for example arise because of programming errors or modeling simplifications; our paper shows an example. In this paper, it will be shown how a truly agent-based representation of learning makes all this considerably more robust.

An additional point is that for large scenarios it is not possible to compute all possible options, and some usually heuristically selected subset needs to be used. In the discrete choice approach, it would be the discrete choice module that selects that subset. In our approach, the agents themselves determine that subset. Depending on the implementation, the difference may not be so large. However, in view of history-dependent learning, for example with mental maps (e.g. (*42*)), the agent-based representation also seems to be more robust here.

4 THE MODULES

The modules which are important for this study are the traffic micro-simulation, the route generator ("router"), and the feedback mechanism, which controls the interaction between the micro-simulation and the router.

4.1 Queue Micro-Simulation

As a traffic micro-simulation we use an improved version of a so-called "queue simulation" (*43*). The improvements are an implementation on parallel computers, and an improved intersection dynamics, which ensures a fair sharing of the intersection capacity among incoming traffic streams (*26*). The details of the traffic simulation are not particularly important for this paper; we expect many traffic simulations to reproduce similar results. The important features of our simulation are:

- Plans following. The feedback framework generates individual route plans for each individual vehicle, and the traffic simulation needs to have travelers/vehicles which follow those plans. This implies that the traffic simulation needs to be microscopic, that is, all individual travelers/vehicles are resolved. Beyond that, however, it does not prescribe the dynamics; everything is possible from smooth particle hydrodynamics (e.g. (*20*, *21*)) to virtual reality microsimulations (e.g. (*22*)).

- Computational speed. We need to run many simulations of 24-hour days — usually about 50 *for a single scenario*. This means that a computational speed of 100 times faster than real time on a road network with several thousands of links (road segments) and several millions of travelers is desirable. Our queue simulation demonstrates that this is feasible.

- Congestion build-up and queue spillback. Although this is not a requirement for the framework in general, the results of the present paper depend on the fact that congestion normally starts at bottlenecks (i.e. where demand is higher than capacity), but then spills backwards into the system and across intersections. Once such congestion is there, it takes a long time to dissolve. The model

should reflect this, *and* it should reflect the physical space that the queued vehicles occupy in the system.

4.2 Router

In addition, we need a router, i.e. a module that generates paths that guide vehicles/travelers through the network from a given origin to a given destination. In addition, the vehicles/travelers have starting times, and the router needs to be sensitive to congestion so that it tends to avoid congested links.

The router we have used for the present study is based on Dijkstra's shortest-path algorithm, but "shortness" is measured by the time it takes an agent to travel down a link (road segment) in the network. 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 trip, 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. (*44*)).

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 taken from a run of the traffic 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. 9am and 9:15am will contribute to the average link travel time during that time period.

4.3 Feedback

Finally, we need the feedback mechanism to couple the router and the traffic simulation. Initially, we feed the traffic simulation with plans based only on free speed travel times. Every time a traffic simulation run completes, the router uses the simulation output to update the travel-time (cost of utilization) associated with each link in the network. After the router updates its view of the network, it generates new plans for a subset (typically a randomly selected 10%) of the drivers, and the entire updated and merged plan-set is fed back into the micro-simulation for another run. We repeat this process as many times as necessary (about 50) until the system "relaxes". Relaxation is as of now not measured by a quantitative criterion, but via judging visualizer output. This will eventually change.

Figures 1(a) and 1(b) illustrate the improvement in the system due to this iterative scheme. Each figure shows a snapshot of vehicle positions in the Gotthard scenario, described in Sec. 5. Figure 1(a) shows a snapshot during the initial (0th) iteration, three hours after the simulation started. Congestion is not known in this iteration, so each traveler assumes free speed travel times, and chooses a route as if it is the only driver in the network. They all choose the freeways, and do not explore alternative routes. Figure 1(b) shows the same situation, but 49 iterations later. Here, the drivers have taken into account the congestion caused by other vehicles on the roadways, so they use many more routes.

5 THE SCENARIOS

The goal of our work is a full 24-hour simulation of all of Switzerland, including transit traffic, freight traffic, and all modes of transportation. This will involve about 7.5 million travelers, and more than 20 million trips (including short pedestrian trips, etc.). A more short-term goal is a full 24-hour simulation of all of car traffic in Switzerland. For this, we will have about 4.5 million car trips. However, we have not yet met this goal completely; current results refer to simulations of the morning peak only. Our network

consists of 10 564 nodes and 28 622 links. This network is provided by the Swiss transportation planning authorities. Besides the standard attributes for geographical location and length, the links have speed, capacity, and type attributes.

In order to test our modules and our framework, we use a so-called **Gotthard scenario**. In this scenario, 50'000 travelers/vehicles start, with a random starting time between 6am and 7am, at random locations all over Switzerland, and with a destination in Lugano/Ticino. Although this scenario has some resemblance with vacation traffic in Switzerland, its main purpose is to test the congestion dynamics of the micro-simulation, and its interaction with the feedback framework. This will become clear later in the text.

We have a second testing scenario, the **6-9 scenario**, which is larger in scale than the Gotthard scenario, but still smaller than our eventual goal. In this scenario, 1'000'000 travelers/vehicles drive typical morning rush hour trips beginning between 6am and 9am throughout the Switzerland network. This is more realistic than the Gotthard scenario, as it is based on real-world trip data, and can be compared to actual road count data taken throughout Switzerland. The actual comparison is reported elsewhere (*45*). More details about the origin of the input data can be found in (*46*).

6 LINK TRAVEL TIME FEEDBACK

Even within the framework as described above, there is considerable flexibility in how to interpret the different pieces. One of these pieces is how to aggregate the link travel times: While the traffic simulation generates link entry and exit times for each individual vehicle, the router needs link traversal times as a function of link entry time. As pointed out above, these latter times also need to be aggregated in order to reduce computational overhead.

One issue is whether to use link entry or link exit times as the basis for aggregation. The way the router works, one would like the average travel time of all vehicles *entering* the links during a specific period of time. In terms of simulation logic, this is awkward since one needs to keep information about when the last vehicle belonging to such a batch has actually left the link.

This issue can be addressed by "backdating" (*47*), that is, one calculates the respective link entering times. For example, assume that the average link travel time for vehicles exiting between 9 and 9:15 is 10 min, and the average link travel time for vehicles exiting between 9:15 and 9:30 is 15 min. By backdating, one would arrive at the result that all vehicles entering between 8:50 and 9:05 need 10 min, and all vehicles entering between 9:00 and 9:15 need 15 min. This clearly leads to gaps and overlaps; piece-wise linear functions can be used to interpolate between the periods.

In our approach, we separate the aggregation from the micro-simulation. That is, the micro-simulation is asked to output event information every time a vehicle leaves a link. A post-processing step then aggregates this data into the information needed by the router.

For the post-processing, we use a pair of AWK scripts; AWK is a standard Unix tool (named for the initials of its authors) that parses line-oriented input files while it also is a fully programmable, C-like interpreted programming language. The first script reads the events file produced by the micro-simulation, filters the events that relate to vehicles entering and exiting links, and compiles them together into an intermediate file that lists, for each vehicle, the time it entered and exited each link on its route. The second script aggregates the output of the first script, to determine the average travel-time on the links.

6.1 Basic feedback

We ran the above setup with the Gotthard scenario. In this section we present the results of that simulation.

For the following, we concentrate on an about 50 km $\times$ 100 km section north of Lugano. For better exposition, the orientation of the plots will be rotated by 90 degrees, so that Lugano now is to the right and the Alps are to the left. Fig. 1(c) shows snapshots of the situation at 19:00 and at 20:00. These and all other snapshots are after 49 feedback iterations. In general, the vehicles are jammed up because there are bottlenecks inside Lugano.

The implausible feature of these plots is that there are traffic jams on the side roads while the freeway is empty. Note that there is no en-route replanning, and so the plan-following vehicles are stuck with their plans for the whole duration of their trips.

The problem was caused by the fact that the router will not react "fast enough" if traffic is moving well at the beginning of the time bin, but not at its end. Cars that enter a link at the beginning of the time bin will leave sooner than the router expects, but those placed at the end of the time bin will leave later than expected.

As an example, suppose that the router is considering routing two agents A and B on a link L during the time bin from 7:00 to 7:15. Suppose further that L is close to free-flowing at 7:00, but is congested by 7:15. Also assume that its average travel-time during this time bin is calculated to be five minutes. If agent A starts out on the link near the beginning of the time bin, say 7:03, it has a clear ride and will be off the link after, say, four minutes. If agent B starts out on the link closer to the end of the time bin, say 7:10, it gets into that congestion and has a longer travel time, say nine minutes. The result is that in the simulation agent A will be minute ahead of the router's schedule, while B will be four minutes behind schedule.

Overall, there are four cases:

1. Congestion building up, and vehicle early in time bin. Then the vehicle will be faster than the router thinks. Since congestion is just building up, it will also be faster in other parts of the system, thus amplifying the initial error.

2. Congestion building up, and vehicle late in time bin. Then the vehicle will be slower than the router thinks. The vehicle will fall behind, and since congestion is building up, it will fall behind further in other parts of the system, thus amplifying the initial error.

3. Congestion going away, and vehicle early in time bin. Then the vehicle will be slower than the router thinks. By falling behind it will encounter less congestion, which will limit how much it falls behind.

4. Congestion going away, and vehicle late in time bin. Then the vehicle will be faster than the router thinks. By being faster it will encounter more congestion, which will limit how far ahead of schedule it is.

From this description it is clear that in particular the first two cases are a problem since the dynamics tends to amplify the errors. In order to test our hypothesis, we describe two modifications to the router in the following.

6.2 Offsetting the Time Bins

As a first possible remedy, giving the router an "early warning" about impending congestion build up is considered. This is done by simply offsetting all the time bin data so that it is ahead of reality by one bin. This causes the router to use the 7:15–7:30 time bin information when it enters a link between 7:00 and 7:15. That way, it will instruct agents to avoid congested links *before* those links actually get congested. It

will also place more vehicles on links that are undergoing transitions from congested to free-flowing at an earlier time. Based on the reasoning above, however, this situation should not cause too much of a problem. – It turns out that this alone is not sufficient: The system still allows the freeways to empty earlier than the side roads (not shown).

6.3 Maximum vs. Average Travel-Time

Another issue with the data used by the router is that it is an *average* of the travel times experienced by the vehicles. As stated above, if the router under-predicts the travel-time for an agent on a link during a time bin, that agent will be behind schedule. Instead of giving the router an early warning, we alter the router's view of the links so that it pays more attention to the travel times of those vehicles who experienced congestion on the links. In other words, we bias the link selection against congested links. The simplest way to do this is to take the *maximum* travel-time experienced on each link during each time bin, rather than the average. – Also this measure does not fix the problem: The freeway still empties earlier than the side roads (not shown).

6.4 Combining Maximum Travel-Time and Offset Time Bins

Neither offsetting the travel times data, nor biasing it toward the maximum reported travel time make the results plausible. As a next test, a combination of the two was tried. That is, the system takes the maximum travel times instead of the averages, *plus* it offsets the resulting travel times data by one bin. Figure 1(d) shows the output from this result. As one can see, the side roads finally empty out before the freeway does, as is plausible.

6.5 Conclusion

After enough analysis, "combining maximum travel-time and offset time bins" finally solved the problem. One essentially had to greatly exaggerate the router's view of the links undergoing transition from free-flowing to congested regimes, so that it could react in time to move travelers away from those links to avoid the congestion. This solution was tailored for this specific problem, however. If there are other routing problems that we discover at a later time, we may have to adjust our travel time reporting strategy again. Such adjustments could conflict with the current method, bringing back the problem of empty freeways with congested side roads. For example, we never looked at routes for vehicles which are early within a time bin, the first of our four above cases. We would like a more robust solution, which can work even if flaws exist in the router or the feedback system.

In the next section we present an alternative solution to the maximum and/or offset strategies, which moves away from adjusting the travel times reporting, to adjusting the behavior of the travelers.

7 THE AGENT DATABASE

7.1 Concept

In the above methods, all agents forgot their previous plans when new ones were created, on the assumption that the new ones were always better than the old ones. But, if the router is flawed, or not obtaining the proper information, this might not (always) be true. So, we now give the agents a *memory* of their past plans (also called decisions or strategies), and the outcome (performance of plans) of those decisions. We allow them to choose their new route plan based on the performance of the route plans in their memory. New, untested routes from the router iteration are given top priority, but if an

agent has tried all of his/her plans before, then he/she chooses one by comparing their performance values. For an early version of this approach, see (*48*).

By giving the agents a memory, we must give them a way to select remembered routes. For a given plan — as a whole — we can find the total time taken to traverse the route. This will be a measure of the performance of the route. This is also called the "score" of the route strategy. Agents can compare scores of remembered routes, and choose one based only on performance information, without knowing anything else about the routes.

The idea here is that we do not need to fix the router to be perfect, as long as it generates reasonable routes most of the time. We can use the original router and travel time reporting strategies, and still get behavior that makes sense. An additional significant advantage is that with our system, other performance criteria besides travel time are easily implemented – one just needs to change the function that calculates the score. In order for that change to be meaningful, one needs a router that generates at least some routes which are "good" under the actual choice criterion that is implemented.

7.2 Implementation of the Agent Database

We introduce a database into the iteration framework to give the agents memory of their plans. Currently, this database is implemented in MySQL, an open-source relational database system (see www.mysql.org; SQL stands for "Standard Query Language," and is the language used to exchange data with the database server). Each time the router generates a new plans file, those plans are added to the database, along with the identifying number of their corresponding agent, and the starting time of the plan. The database also stores, for each plan, the most recently measured travel time (score) made by the agent for that plan; and a flag that, when true, marks the plan as being the one used by its agent in the most recent micro-simulation. For new, untried plans generated by the router, the travel-time is considered to be zero, and the agent is forced to always choose that plan next. See Fig. 2(a) for an example of how the database stores information. At this stage, the database allows each agent to choose a plan (explained in more detail below), and updates the flags to reflect the new choices. These chosen plans are output to the micro-simulation for another run.

After a simulation run is finished, its output is processed into link travel times for the router, and a file containing each agent's trip time. The latter is read into the database, where it is stored together with the corresponding route-plan. At this point, the database is ready for the next iteration, when the router will again generate a new set of plans that must be entered into the database.

7.3 How Plans Are Chosen Based on Performance

An important detail left out of the above explanation is how the performance (total travel-time) information is used by the agents to choose their plan for the next iteration. Each agent a uses a logit model (*16*) to select option i:

$$P(T_{a,i}) = \frac{e^{-\beta \cdot T_{a,i}}}{\sum_j e^{-\beta \cdot T_{a,i}}} \,, \tag{1}$$

where $T_{a,i}$ is the travel time.

The value of β determines how likely it is that a "non-best" plan will be chosen. For the Gotthard scenario, we chose the value of β so that about 90% of the agents would choose their best possible plan. The corresponding value of β was $1/(360\ \mathrm{sec})$.

7.4 Results of Agent Database on the Gotthard Scenario

Figure 1(e) shows the results of using the original feedback method from Sec. 6.1 plus the agent database, with plans selected as described above. As one can see, the freeway problem is avoided when the agents have memory of their plans, *without any other changes in the router or in the travel time feedback*. If the router starts putting too many agents on the side roads, some will eventually try out an old plan that used the freeway and find that it has a good performance, so will likely use that plan again. As long as they remember one or more plans that utilize the freeway, the agents can decide for themselves to use it, bypassing the side road choice of the router. Thus, the agent database gives an added flexibility and robustness to the system, so that even with a flawed router or feedback mechanism, the results come out satisfactorily. This value of β we chose seemed to work well, but future work will likely need to explore the outcome of other values for this constant.

7.5 Computational Speed of the Agent Database

Figure 2 shows several graphs relating to the execution time as a function of iteration number for the important steps in the agent database feedback approach.

Besides for the Gotthard scenario, the agent database was also used for simulations of the 6-9 scenario, as explained in Sec. 5. Further information about this, including a comparison to static assignment results and to field data volume counts, are presented elsewhere (*45*). The computational performance results for that study are included here.

Figure 2(b) shows the execution time for those feedback steps that are linear in time based on their input size ($\mathbb{O}(n)$). The number of plans in the database goes up by 10% each iteration, and the execution time of these operations is going up linearly with the iteration number (and thus the number of plans). So, we know that most of the database operations scale up (essentially) linearly. This graph also allows one to compare the results of the two test-cases, based on their problem size. The Gotthard scenario began with 50'000 plans while the 6-9 scenario began with 1'000'000. By multiplying the Gotthard execution times by 20, one can see those lines match with the 6-9 lines, further supporting linear complexity.

Figure 2(c) shows some other operations (such as parsing the events files) that do not have linear execution time, but mostly constant execution time ($\mathbb{O}(1)$). These rely on the amount of data produced by the simulator, such as the number of events.

Figure 2(d) shows the execution time (for the 6-9 scenario only) of different versions of the *output-travel-times* operation, which requests the output of an agents' travel-time scores. In the "old" method, *output-travel-times* asked the database server to sort the information by the agent and plan number. This took a long time within the database ("output-tt w/DB sort"). This was improved significantly by asking the database for unsorted data and then sorting it externally (via the Unix "sort" command). The corresponding execution times for database output and for external sorting are "output-tt unsorted" and "sort-tt". Even when adding up those two time contributions ("output-tt + sort"), they are still significantly less than with the original approach.

Figure 2(e) summarizes the other three graphs by depicting the contribution of each operation to the total execution time of each iteration. In this figure, the results for the Gotthard scenario are omitted for clarity. The main point of this figure is that, excluding the microsimulation, the feedback process takes an average of about 2600 seconds (about 43 minutes) to execute for about 1 million agents.

8 DISCUSSION

The true potential of multi-agent simulations in the area of transportation science has not yet been fully tapped. An important point of a true CAS method is that each agent

has several different individual strategies, and that learning methods are applied to generate new strategies, either via crossbreeding of existing strategies or via innovation. However, virtually no existing (large scale) implementation allows for multiple strategies per agent. Even TRANSIMS, which is based so much on individual intelligent agents, in its default configuration does not exploit the potential of multiple strategies per agent, although the design would allow it.

An open issue concerns the calibration and validation of agent-based techniques. There are several related but separate issues:

- **Verification** that a code corresponds to its specifications. It has been our experience also in other projects such as in climate simulations that this goal is difficult to achieve in practice. Also, as one has seen in this paper, even code fully corresponding to specifications can give implausible results. Formal proofs of correctness have now become possible for medium-sized projects (B. Meyer, personal communication), but are relatively expensive and possibly incompatible with a research environment. Still, some process of verification and "code freezing" should be eventually implemented.
- **Calibration** means that the parameters of the model were adjusted so that they match some given set of data as well as possible. In our case, the microsimulation is (by definition and via some testing) fully calibrated against the input data that it uses. The routing model uses the normatively declared time-dependent fastest path. And the feedback mechanism uses a heuristic 10% learning rate, which yields fast relaxation but has no additional justification.
- **Validation** means that the calibrated model is used for some real world problem and compared to some field data, preferably against field data *that has not been used for the calibration*. We have in fact done such a study for traffic in Switzerland, where realistic OD matrices were fed into our system and the resulting volumes for the morning rush hour were compared against reality. The details of this go beyond the scope of this paper and can be found in Ref. (*45*). The overall result was an average relative error of less than 26%. This was better then the results of an assignment model that was used for comparison, and interestingly this result came out *although the OD matrices were calibrated to optimize the assignment result against the counts*.

In general, it is our belief that validation of agent-based models should be in the field, not on synthetic or reduced scenarios. A good way, in our view, would be to have international competitions as they are common in other fields of science. Such a competition would be organized around major infrastructure changes. It would give access to all possible input data to the scenario, the predictions would be submitted *before* the infrastructure change is executed, and after the infrastructure change the predictions would be checked. Although each individual competition would have a strong random component, one would expect that in the long run the better methods would produce the better results.

9 CONCLUSION

The purpose of this paper and this study is to demonstrate that for multi-module transportation simulations, not only is the functionality of the single modules important, but also how they interact. In particular, an agent-based implementation of the interfaces between the modules is capable of correcting for artifacts in the modules. An agent-based representation means that travelers are considered as agents, which have a memory of different strategies and their respective performances. In general, they chose the strategy with the best performance, but from time to time re-try one of the other strategies just to check if its performance is still unchanged. Also from time to time, new strategies are generated and added to the pool.

In this particular example, we apply this approach to route feedback for dynamic traffic assignment. The problem was that the router uses aggregated feedback information from the micro-simulation, and that this aggregation with most plausible algorithms lead to artifacts in the resulting traffic. Specifically, the router under-estimated long distance travel times, leading to the fact that the router assumed the existence of congestion for later parts of the trip while in fact the congestion was long gone. This resulted in travelers using the side roads where the freeway would have been much better. The use of the agent data base solves this problem *without any changes in the router*. That is, even when the router consistently generates faulty plans, the agent database approach will compensate for this as long as at least some of the routes are plausible. The approach was implemented using MySQL as a data base, and AWK as scripting languages. The agent database was then applied to a simulation of the morning rush hour of "all of Switzerland". The results of this are reported elsewhere (*45*).

ACKNOWLEDGMENTS

We would like to thank the Swiss regional planning authority (Bundesamt für Raumentwicklung) and Milenko Vrtic at the Institute for Transportation Planning (IVT) of ETH Zürich for providing the Switzerland network; Andreas Völlmy for the Gotthard scenario initial plan set data; and Nurhan Cetin for the parallel queue simulation. Marc Schmitt does an excellent job in maintaining our computer system.

This work was funded by ETHZ core funding and by the ETHZ project "Large scale multi-agent simulation of travel behavior and traffic flow".

REFERENCES

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

[2] K. Nagel. Distributed intelligence in large scale traffic simulations on parallel computers, submitted. See sim.inf.ethz.ch/papers.

[3] P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson. Microsimulation of urban development and location choices: Design and implementation of UrbanSim. *Networks and Spatial Economics*, 2003. In press.

[4] S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[5] T. Kohonen. *Self-organizing maps*. Springer Series in Information Sciences, 30. Springer, 2000.

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

[7] M. Dorigo, G. DiCaro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[8] R.G. Palmer, W. Brian Arthur, J. H. Holland, Blake LeBaron, and Paul Tayler. Artificial economic life: a simple model of a stockmarket. *Physica D*, 75:264–274, 1994.

[9] J. Ferber. *Multi-agent systems. An Introduction to distributed artificial intelligence*. Addison-Wesley, 1999.

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

[11] SWARM. www.swarm.org.

[12] RePast. Recursive porous agent simulation toolkit. repast.sourcefourge.net.

[13] D. Lohse. *Verkehrsplanung*, volume 2 of *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung*. Verlag für Bauwesen, Berlin, 1997.

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

[15] In D. Hensher and J. King, editors, *The Leading Edge of Travel Behavior Research*. Pergamon, 2001.

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

[17] J. L. Bowman. *The day activity schedule approach to travel demand analysis*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[18] S. Ramming. *Network Knowledge and Route Choice*. PhD thesis, MIT, MA, USA, 2002. See web.mit.edu/sramming/www/Thesis/ramming.defense.draft.pdf.

[19] H. A. Rakha and M. W. Van Aerde. Comparison of simulation modules of TRANSYT and INTEGRATION models. *Transportation Research Record*, 1566:1–7, 1996.

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

[21] DYNASMART. See www.dynasmart.com. Also see dynamictrafficassignment.org.

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

[23] C. Gawron. *Simulation-based traffic assignment*. PhD thesis, University of Cologne, Germany, 1998.

[24] G. D. B. Cameron and C. I. D. Duncan. PARAMICS — Parallel microscopic simulation of road traffic. *J. Supercomputing*, 10(1):25, 1996.

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

[26] N. Cetin and K. Nagel. Parallel queue model approach to traffic microsimulations. Paper 03-4272, Transportation Research Board Annual Meeting, Washington, D.C., 2003. Also see sim.inf.ethz.ch/papers.

[27] Gürling T. and Young W. Perspectives on travel behavior: Decision paradigms. *Travel Behaviour Research, The Leading Edge*, pages 219–225, 2001.

[28] R. Selten and M. Schreckenberg. Experimental investigation of day-to-day route choice behavior. See sim.inf.ethz.ch/papers.

[29] D. Helbing, M. Schonhof, and D. Kern. Volatile decision dynamics: experiments, stochastic description, intermittency control and traffic optimization. *New Journal of Physics*, 4(33), 2002.

[30] David E. Kaufman, Karl E. Wunderlich, and Robert L. Smith. An iterative routing/assignment method for anticipatory real-time route guidance. Technical Report IVHS Technical Report 91-02, University of Michigan Department of Industrial and Operations Engineering, Ann Arbor MI 48109, May 1991.

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

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

[33] R. Cayford, W.-H. Lin, and C.F. Daganzo. The NETCELL simulation package: Technical description. California PATH Research Report UCB-ITS-PRR-97-23, University of California, Berkeley, 1997.

[34] T. Schwerdtfeger. *Makroskopisches Simulationsmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO)*. PhD thesis, University of Karsruhe, Germany, 1987.

[35] Q. Yang. *A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems*. PhD thesis, Massachusetts Institute of Technology, 1997. See also its.mit.edu.

[36] VISSIM. Planung Transport und Verkehr (PTV) GmbH. See www.ptv.de.

[37] T. L. Friesz, D. Bernstein, N. J. Mehta, R. L. Tobin, and S. Ganjalizadeh. Day-to-day dynamic network disequilibria and idealized traveler information systems. *Operations Research*, 42(6):1120–1136, 1994.

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

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

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

[41] M. Rickert and K. Nagel. Issues of simulation-based route assignment. Presented at the International Symposium on Traffic and Transportation Theory (ISTTT) in Jerusalem, 1999.

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

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

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

[45] B. Raney, N. Cetin, A. Völlmy, M. Vrtic, K. Axhausen, and K. Nagel. An agent-based microsimulation model of Swiss travel: First results. Paper 03-4267, Transportation Research Board Annual Meeting, Washington, D.C., 2003. Also see sim.inf.ethz.ch/papers.

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

[47] I.R. Porche. *Dynamic traffic control: Decentralized and coordinated methods*. PhD thesis, University of Michigan, 1998.

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

List of Figures

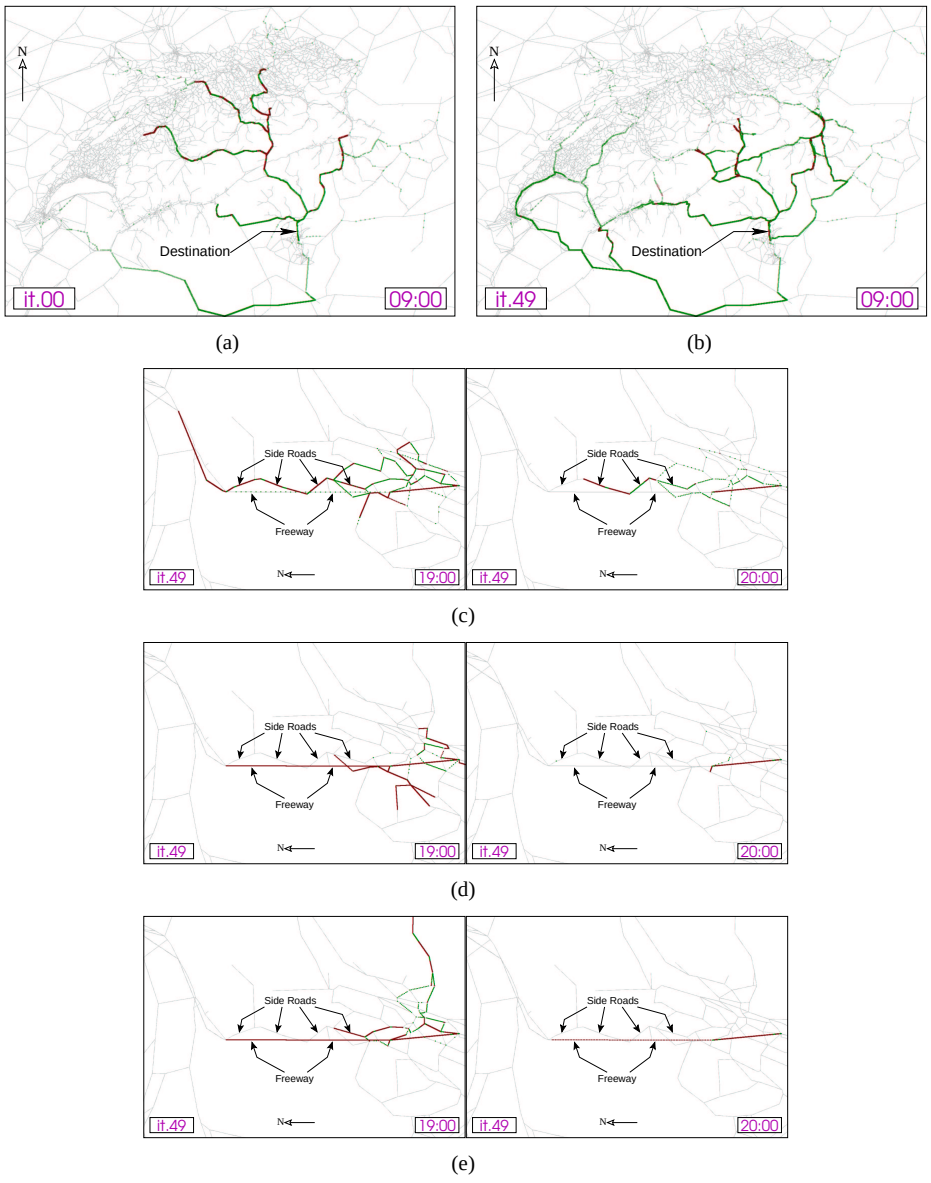


FIGURE 1: ***(a)–(b):*** **Example of relaxation due to feedback.** ***(a)*** **Iteration 0 at 9:00 — all travelers assume the network is empty.** ***(b)*** **Iteration 49 at 9:00 — travelers take more varied routes to try to avoid one another.** ***(c)–(e):*** **A freeway and side roads with the different travel time feedback method at 19:00 (left) and 20:00 (right).** ***(c)*** **Original method. The side roads contain many vehicles while the freeway contains very few or none.** ***(d)*** **Combined offset time bins with maximum travel time strategy. The side roads are finally empty, while the freeway now contains vehicles. This is what is expected from the scenario.** ***(e)*** **Agent database method. As with method (b), the side roads are emptying, while the freeway contains vehicles. This shows the agent database is a robust solution to the freeway problem.**

plans table:

agent	plan_num	is_new	start_time	plan
1	1	0	25200	<text string 1>
1	2	1	25200	<text string 2>
2	1	0	25380	<text string 3>
⋮	⋮	⋮	⋮	⋮

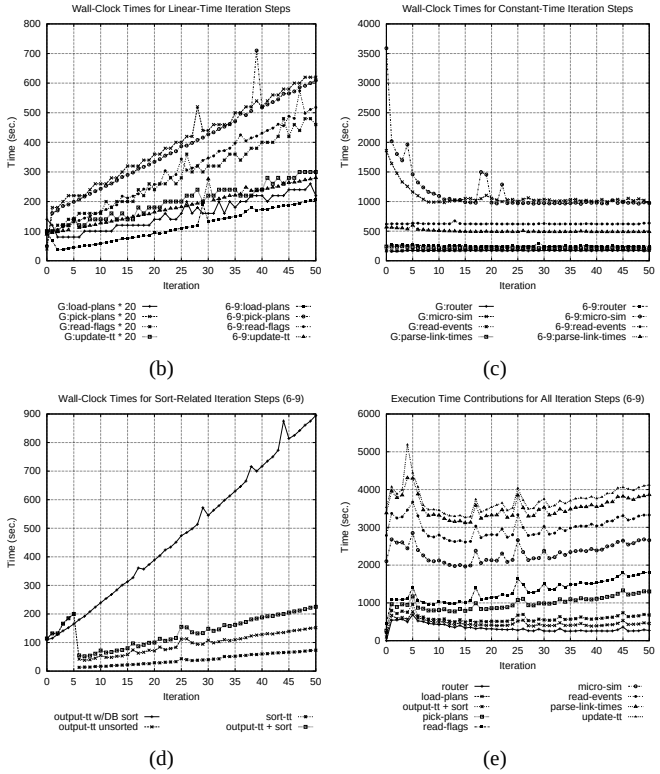
travel-times table:

agent	plan_num	travel_time
1	1	462
1	2	0
2	1	1047
⋮	⋮	⋮

flags table:

agent	plan_num	flag
1	1	1
1	2	0
2	1	1
⋮	⋮	⋮

(a)



(b) (c) (d) (e)

FIGURE 2: ***(a)* Example tables in the agent database. The "agent" and "plan_num" fields are combined into the primary key for all three tables. The "plan" field of the plans table contains a text string consisting of link identifiers and other information that the router outputs.*(b)* Execution times of the linear-time feedback steps. *(c)* Execution times of the constant-time feedback steps. *(d)* Execution times of *output-travel-times*, with internal and external sorting. *(e)* Execution time contributions of all steps.**