A first approach to a continuous simulation of daily travel

Fabian Märki

David Charypar

Kay W. Axhausen

STRC 2010 **September 2010**

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

STRC 2010

A first approach to a continuous simulation of daily travel

Fabian Märki
IVT
ETH Zurich
CH-8093 Zurich
phone: +41-44-633 33 25
fax: +41-44-633 10 57
fabian.maerki@ivt.baug.ethz.ch

David Charypar
IVT
ETH Zurich
CH-8093 Zurich
phone: +41-44-633 35 62
fax: +41-44-633 10 57
charypar@ivt.baug.ethz.ch

Kay W. Axhausen
IVT
ETH Zurich
CH-8093 Zurich
phone: +41-44-633 39 43
fax: +41-44-633 10 57
axhausen@ivt.baug.ethz.ch

September 2010

Abstract

This paper introduces a microscopic traffic simulation that continuously simulates activity-based agent behavior and the resulting traffic. It drops iterative optimization, that builds on stochastic user equilibria, and moves to a continuous planning approach. The behavioral model of this approach utilizes the concept of needs to model continuous demands. Several intuitive parameters control demand and facilitate calibration of versatile behaviors. These behaviors originate from a planning heuristic which makes *just in time* decisions about upcoming activities an agent should execute. The planning heuristic bases its decisions on the current need levels of an agent and the development of these levels in the near future. We illustrate the model through simulation runs and suggest directions of future research.

Keywords

need based planning, continuous activity generation, microscopic traffic simulation

1 Introduction

Microscopic travel demand simulation software (e.g. Balmer (2007)) simulates people individually. Each virtual person, referred to as agent, specifies day-plans which consist of activities. These activities are then executed with a simulation software. Such simulation runs provide agents with feedbacks about the utility of their day-plans. The aim of each agent is to maximize its utility and as a consequence, they adapt their day-plans according to the previous simulation results. This replanning step is repeated until the simulation reaches a stochastic user equilibrium where travel demand and travel supply are consistent (Nagel and Flötteröd (2009)).

The design of the above-mentioned approach leads to high computational complexity which limits the simulation horizon of standard size scenarios to a single day. This makes it difficult to investigate effects that occur between days or between weeks. Another limitation is that agents have to commit themselves to a specific day-plan. This restricts the ability of agents to react to unexpected events. Even worse, the repetitive nature of the simulation provides agents with the knowledge of an unexpected event. Accordingly, agents adapt their day-plans in foresight of that event which is impossible in real life. These shortcomings keeps us from utilizing this approach for our investigations. As a consequence, a different simulation becomes necessary.

We propose a microscopic travel demand simulation which is capable of modeling demand continuously. We use a behavioral model which utilizes the concept of needs to model the continuous demand. Agents continuously track their need levels and provide this knowledge to a planning heuristic which makes decisions about upcoming activities agents should execute. This makes it possible, that agents can spontaneously react to unexpected events. At the same time, it also reduces memory consumption because agents do not need to keep track of day-plans. We present different calibration mechanisms which extend the need-based approach to enable distinct behavior based on the day of the week or other constraints like public holidays. These extensions make the need-based approach applicable to model not only recurrent activities but also the richer contents of everyday life. We plan to take advantage of developments in distributed computation to decrease computation time further. The proposed activity planning module will be embedded in a distributed computation environment (Charypar *et al.* (2010)). This helps to overcome the burden of long computation times but also introduces constraints to the model. We illustrate how we respect these constraints and appropriately design our algorithms.

The remainder of this paper is structured as follows: First, we discuss the dynamics of the need-based behavior. We then introduce the structure of our need-based model and its calibration mechanisms. The next section describes the planning heuristic and its key features. This is followed by an explanation of the model calibration and its validation on several examples. We conclude the paper with an outlook on coming tasks.

2 Other Work

Arentze and Timmermans (2006) introduce need-based theory and propose a model for activity generation (Arentze and Timmermans (2009)) which assumes that utilities of activities are a dynamic function of needs. Whereas Arentze and Timmermans use needs to generate day-plans for a multi-day planning period, we use needs to make *just in time* decisions about upcoming activities agents should execute. Kuhnimhof and Gringmuth (2009) use a different approach and simulate a 7-day model by recycling existing schedules. They could show that their approach produces realistic day plans but suffers from a certain inflexibility when agents should spontaneously react to unexpected events. Charypar and Nagel (2006) formulate the planning procedure as a reinforcement learning problem and report that this approach has a poor performance for large scenarios. We try to overcome this problem by using a planning heuristic which approximates the optimal solution. Schlich (2004) considers travel as a consequence of people's endeavor of need satisfaction and Schönfelder (2006) sees the satisfaction of needs as an explanation of the rhythms of individual travel behavior. Charypar *et al.* (2009) address the shortcomings of the existing equilibrium model and propose a need-based activity planning system.

3 Need Driven Behavior

This section introduces the need driven behavior. The first subsection is a lightweight overview and meant for readers who want to get a general understanding of the topic. The second subsection introduces the mathematics use to describe the development of need levels over time.

3.1 Overview

The need-based approach assumes that needs build up continuously over time (Arentze and Timmermans (2009), Schönfelder (2006)). Sooner or later, an increasing need drives an agent to perform an activity to reduce that particular need. For instance, the activity *meet friends* might satisfy a need to *socialize with other people* whereas the activity *grocery shopping* satisfies a need for *food stock*. As a side effect, an activity can also decrease or even increase other needs. The activity *shopping* might have an exhausting effect resulting in a decrease in the need for *physical exercise* while it increases the need for *relaxation*. Accordingly, need levels are the central component to model the state of agents.

Agents use a metric that describes the utility they can gain through the satisfaction of a need by executing an specific activity at a certain location. This utility is defined by the difference between the need levels before and after the agent satisfied the need (see Fig. 1). We make sure that longer activities provide a higher utility by multiplying the resulting utility by the default execution duration of the activity. The default execution duration defines the duration an activity has to reduce a need from a high level to a low level. Including this factor is important since otherwise, agents would always choose activities with the shortest possible duration because this would result in the highest overall utility per time unit.

The need decrease function is of central importance because it not only describes the development of the utility over time but also the activity duration an agent should choose to satisfy the corresponding need. We use an asymmetric S-shaped function (see Fig. 1) with an inflection point as presented by Joh (2004). Feil *et al.* (2010) used this function to describe activity duration and estimated its parameters. This is consistent with our requirements because the necessary time to decrease a need level correlates to the duration an agent performs the corresponding activity. We include an additional parameter which defines the fraction of time an agent was able to execute an activity (we refer to this parameter as *execution rate*). This parameter is of importance if an agent decides to satisfy a need during a time period where the corresponding activity cannot or can only be partially executed (e.g. because the shop closes).

We decided to use the same functional form to describe need increase as we used for the need decrease.

3.2 Mathematical Formulations

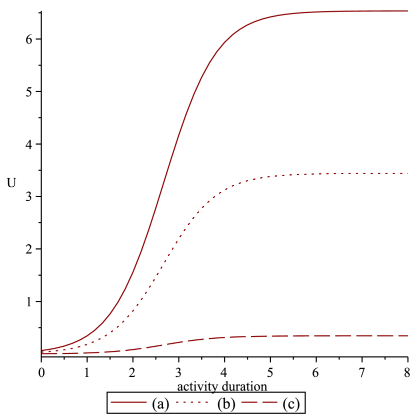
The aim of this subsection is to introduce the mathematics we use to describe the utility and need development over time. We use functions called generalized logistic curves which originate from biological science (Richards (1959)). Joh (2004) introduced this function to travel behavior simulation to characterize utility and Feil *et al.* (2010) reused them to describe activity duration.

The utility an agent i receives for satisfying a need j is defined as (we omit indices i and j for simplicity)

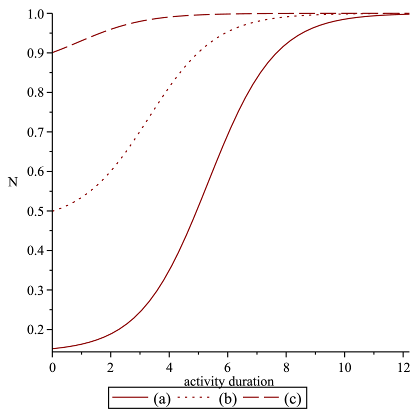
$$U_k(N^t, N^{t+\Delta t}, defActDur_k) = (N^t - N^{t+\Delta t}) \cdot defActDur_k \tag{1}$$

Figure 1: Function plots showing the development of utility and need levels

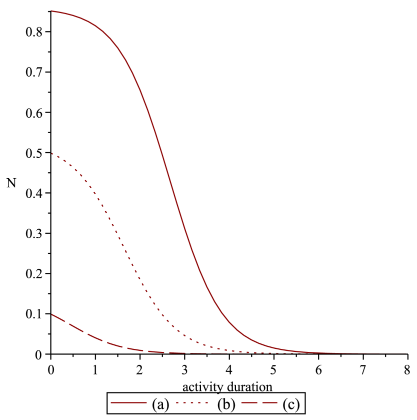
(a) Utility gain for activities with different default durations



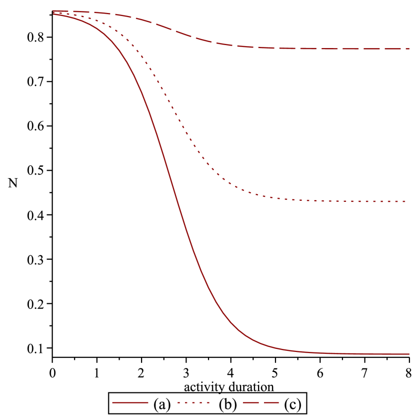
(b) Need increase for different start levels



(c) Need decrease for different start levels



(d) Need decrease with different execution ratios



where Δt is the duration agent i decided to satisfy need j. N^t represents the need level at the time agent i starts satisfying need j and $N^{t+\Delta t}$ is the need level afterwards. $defActDur_k$ is the default duration activity k of agent i should be executed to satisfy need j. Including this parameter ensures that activities with a longer default duration result in higher utility than activities with a shorter default duration.

The requirements for the need decrease function include, that the need should decrease monotonically starting from the need level N^t that was reached before the agent decided to satisfy the need and that the need level should stay above a predefined minimum need level. The decrease an agent i experiences for its need j is defined as (we omit indices i and j)

$$N_k^{dec}(\Delta t, N^t, execRate_k) = N^t - \frac{N^t \cdot execRate_k}{\left(1 + \gamma_{dec} \cdot exp\left[\beta_{dec}(\alpha_{dec} \cdot \frac{N^t}{N^{max}} - \Delta t)\right]\right)^{1/\gamma_{dec}}} \tag{2}$$

where Δt is the duration agent i satisfies need j. N^t represents the need level at the time agent i starts satisfying need j. $execRate_k$ defines the time quota need j could be satisfied during Δt. This parameter is of importance if an agent decides to satisfy a need during a time period where the corresponding activity k cannot or can only be partially executed (e.g. because the shop closes). N^{max} defines the maximal possible need level. So far, we use the same N^{max} for all needs because this simplifies the comparison of need levels in the calculations of the heuristic. α_{dec}, β_{dec} and γ_{dec} are parameters that influence the shape of the S-curve. We refer readers to Feil *et al.* (2009) for a detailed explanation of the properties of the function.

The requirements for the need increase function include, that the need should increase monotonically starting from the need level N^t that was reached after the need was satisfied the last time and that the need level should stay below a predefined maximum need level N^{max}. We decided to use the same functional form as we use for the need decrease. The increase an agent i experiences for its need j is defined as (we omit indices i and j)

$$N^{inc}(\Delta t, N^t) = N^t + \frac{N^{max} - N^t}{\left(1 + \gamma_{inc} \cdot exp\left[\beta_{inc}(\alpha_{inc} \cdot \left(1 - \frac{N^t}{N^{max}}\right) - \Delta t)\right]\right)^{1/\gamma_{inc}}} \tag{3}$$

where Δt is the elapsed time since agent i satisfied need j for the last time and N^t represents the need level at that time.

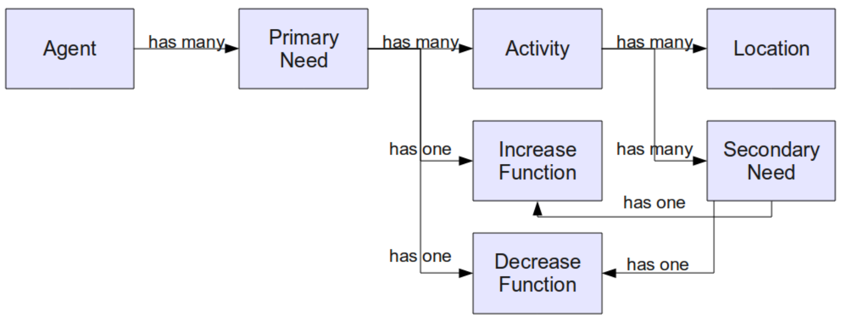
4 Definition of Need Based Model

This section outlines the model of our continuous activity planning module. This includes a discussion of its calibration mechanisms and its relevance for a simulation of weekly and monthly patterns in a distributed computation environment.

4.1 Model Definition

The central component of our model (see Fig. 2) is the *Agent*. Each *Agent* can have several *Primary Needs* assigned which it has to satisfy. These *Primary Needs* can be needs which are only assigned to this specific agent or they can also be needs that are shared e.g. with household members (see below). *Increase Functions* and *Decrease Functions* describe the need levels of *Primary Needs* over time (examples of realizations of these functions are provided in Fig. 1). Each *Primary Need* can be satisfied by performing several *Activities*. For instance, a need for *social contact* can be satisfied by activities like *meet friends*, *go clubbing* or *talk to relatives*. An

Figure 2: Components of our model and their interdependencies



Activity can be performed at several *Locations*. Performing an *Activity* can have a side effect on other needs (Arentze and Timmermans (2009)). We take care of this effect by assigning *Secondary Needs* to an *Activity*. *Secondary Needs* refer to *Primary Needs* and change their need levels through *Increase* and *Decrease Functions*.

4.2 Behavior Calibration Mechanisms

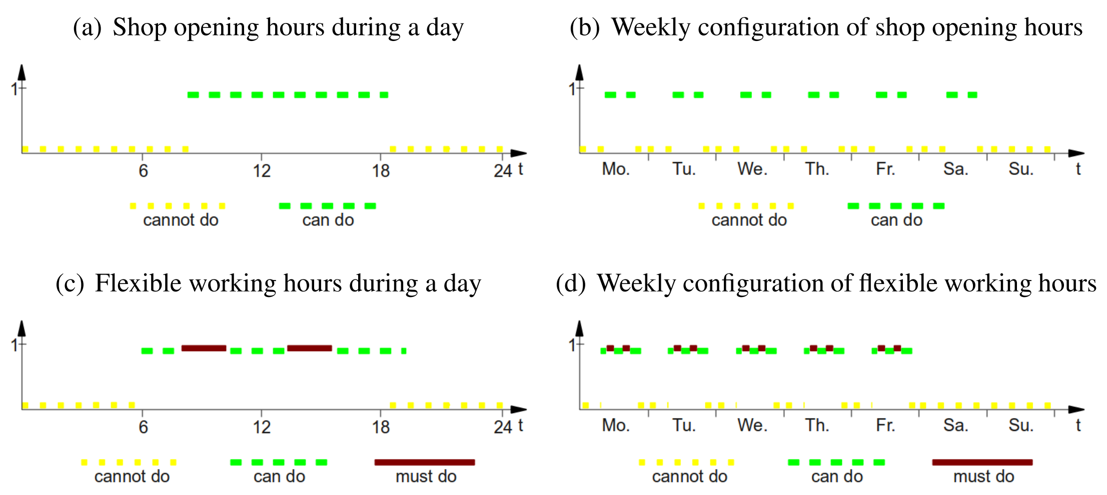
The need for sleep is an excellent example for a recurrent need because it has a periodic occurrence and happens every day more or less at the same time independent of the day of the week. When it comes to needs which have distinct behavior depending on the time of day or the day of the week, the need-based approach experiences difficulties to model behavior appropriately. For instance, an agent can satisfy its need for *grocery shopping* only during opening hours and the need for *work* during weekdays. Similar difficulties become apparent when needs have different increase rates depending on the time of day or the weekday. For instance, the need for *food* grows differently during daytime than at night and the need for *work* shows a different behavior during the week than at the weekend. As a consequence, an enhancement of the model becomes necessary to make it suitable to model true life patterns. Consequently, we enhance the model with two additional calibration mechanisms:

- A mechanism that tells agents when they cannot, can or even must satisfy certain needs,
- A mechanism that facilitates different grow rates of needs depending on the time of day and the day of the week.

4.2.1 Execution Period

Execution periods define time durations during which an agent can satisfy a specific need. With the parameters *can do* and *cannot do* it is, for instance, possible to model shop opening hours

Figure 3: Execution periods modeling shop opening hours and flexible working hours



(see Fig. 3). Through the additional parameter *must do* it is possible to model more complex contexts. Figure 3 also shows an execution period which models flexible working hours with periods where the agent cannot work, periods when it can work and periods during which its presence at the work place is required.

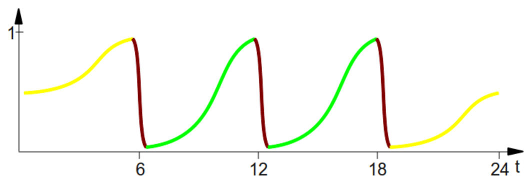
These execution periods can be assigned to locations. This makes it possible to precisely specify at which location an agent has to execute which activity. This is a powerful feature because it provides a simple tool to model appointments of agents through a *must do* execution period. Through this feature, agents can agree on a meeting place and meeting time and by inserting a *must do* execution period in their memory, the planning heuristic will take care that agents keep their promise and will be on time at the right location to meet their friends. This will especially be convenient for models which consider interactions in social groups. At the same time, it is also possible to extract the execution rate of an activity from an execution period. This value is important for the calculation of the need decrease (see Eq. (2)). The execution rate is obtained through the integration of the actual possible execution period over the total execution time of an activity.

4.2.2 Need Increase Rate

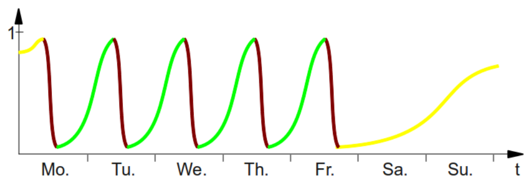
We realized that it can become necessary to model needs with different need increase rates depending on the time of day or the day of the week (see Fig. 4). An example is the need for *food*. In average, we eat three times per day. However, the occasions when we eat are not equally distributed over the day but are accumulated during daytime. A similar problem arises

Figure 4: Different need increase rates

(a) Changing need increase rates depending on the time of day



(b) Changing need increase rates depending on the weekday



when we model the need to *work* over a week (see Fig. 4). During the week, the need regularly increases. However, we do not work during the weekends and accordingly, the need increase rate should be different during weekends. Otherwise, agents would show up at the workplace on Mondays with an immense desire to work. Driven by these insight, we extended the model so that it provides the possibility to define different need increase rates.

4.3 Shared Needs

Our agents base their decisions concerning the next need to satisfy on their need levels and the resulting utility. Since the simulation runs on a distributed environment, it is computationally expensive for agents to communicate and negotiate with other agents about who should satisfy a shared need (e.g. needs that are shared among household members like *grocery shopping*). Consequently, a solution which does not require communication between agents during the decision making process, during which agents can be distributed among different computation nodes, becomes necessary in order to reduce computational costs. The solution we propose is to make sure that agents meet in cyclic intervals, e.g. through a *must do* execution period for household members during bedtime. During this period, agents negotiate about a reallocation of shared needs among each other based on their current *stress level*. Needs that are assigned to an agent become its responsibility. This is a simple and powerful solution because it does neither require a special treatment of shared needs during the decision making process concerning the next need to satisfy nor communication between involved agents.

We define the *stress level* of agent i as the amount of time agent i would need to reduce all its need levels to a moderate level. This is defined by (we omit index i)

$$SL = \sum_{j=0}^{n} N_j^{dec}(N_j^t, N_j^{t+\Delta t})^{-1} \tag{4}$$

where the inverse need decrease function $N_j^{dec}(...)^{-1}$ is summed up over all needs j of agent

i. N_j^t defines the current need level and $N_j^{t+\Delta t}$ represents the predefined need level we want to achieve.

Since every need contributes to the *stress level*, it is of interest to find the combination of needs which leads to an equally distributed *stress level* among the involved agents. The problem of equally distributing needs among agents is related to the knapsack problem. Since we do not require an optimal solution, we utilize a variation of the greedy approximation algorithm proposed by Dantzig (1957). Shared needs are sorted in decreasing order according their contribution to the *stress level*. Thereafter, one need after another is assigned to the agent with the lowest *stress level* until all shared needs are allocated to an agent.

Algorithm 1 Distribute shared needs among involved agents

```
$sortedSharedNeeds \leftarrow getAllSharedNeedsSorted()$
$agents \leftarrow getAllInvolvedAgents()$
for all $need$ in $sortedSharedNeeds$ do
    $agent \leftarrow getAgentWithLowestStressLevel(agents)$
    $agent.add(need)$
end for
```

5 Planning Heuristic

In a continuous simulation, agents cannot replan the same day until an optimal state emerges like Balmer (2007) does in the equilibrium model. Kuhnimhof and Gringmuth (2009) simulated a 7-day model by recycling existing schedules. While this approach produces realistic day plans it struggles with an inflexibility when agents should spontaneously react to unexpected events. Charypar and Nagel (2006) formulated the planning procedure as a reinforcement learning problem and reported that this approach has a poor performance for large scenarios.

We think that a planning heuristic is a feasible approach to overcome the limitations described above. It enables agents to react to unexpected events because they make their decisions on the fly. The aim of an heuristic is to quickly approximate an optimal solution. Consequently, it is also not necessary to have perfect knowledge about the state and plans of other agents as it would be desirable for the optimum seeking equilibrium model. This is helpful since we simulate our agents on a distributed environment where global knowledge only comes with extremely high computational costs which would defeat the primary advantage of a distributed environment. One could argue that people are seeking for optimal day plans. However, other authors (e.g. Simon (1955) and Schlich (2004)) doubt that the behavior of people can be explained as an utility maximization function. Schlich consolidates his doubts by pointing out, that people often have a lack of knowledge of all their possible options and nearly always do not have the time to validate them. Accordingly, we think that using a decision procedure that uses local information to approximate an optimal solution might turn out to be close enough to real world behavior.

5.1 Considered Aspects

The planning heuristic we use in our model builds on the need-based approach and its calibration parameters we described above (see section Definition of Need Based Model). Thereby, it considers following aspects:

- The utility an agent could obtain through the execution of an activity and the implied need satisfaction,
- The flexibility to postpone the satisfaction of a need to a later moment of the current or the next execution window,
- Needs which must be satisfied because of a *must do* execution period.

The aim of the first aspect is to provide agents the possibility to execute activities with a high satisfaction potential. In that respect, our agents are also utility maximizers which consider the immediate future. We obtain the utility through the utility function 1.

The problem of producing good day plans is related to the *Vehicle Scheduling Problems with Time Window Constraints*. Atkinson (1994) and Ioannou *et al.* (2001) investigated this problem in depth and recognized the importance of a look-ahead capability which considers the flexibility to postpone a task. Applied to our problem, this feature ensures that agents look ahead into the future and realize when it is necessary to execute an activity before its need grows to an extreme level. An example of such a situation is the need for *grocery shopping* before a long weekend. If agents do not look into the future and go *grocery shopping* before the long weekend starts, their food stock would be consumed in the middle of the long weekend without an option to restock.

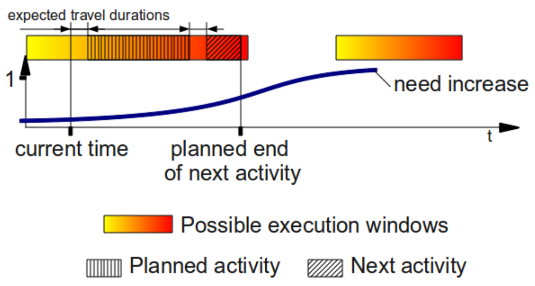
5.2 Heuristic

The planning heuristic utilizes a three step decision procedure. In a first step, it determines the activity-location pair that would yield the highest utility. It uses a measurement which we call *utility density*. *Utility density* is defined as (we omit indices i and j)

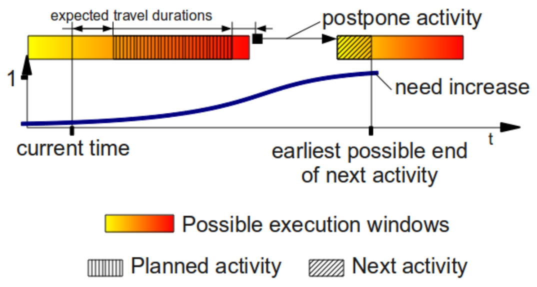
$$UD_{k,l}(U, eTDur_l, eEDur_k, minEDur_k) = \frac{U}{eTDur_l + max(eEDur_k, minEDur_k)} \qquad (5)$$

Figure 5: Illustration of postponement considerations for the current and the next execution window

(a) Agent considers postponing an activity within the current execution window based on an awareness of time pressure


expected travel durations
1
need increase
current time
planned end of next activity
t
Possible execution windows
Planned activity
Next activity

(b) Agent considers postponing an activity to the next execution window based on an awareness of long waiting time until the next execution window starts


expected travel durations
postpone activity
1
need increase
current time
earliest possible end of next activity
t
Possible execution windows
Planned activity
Next activity

where U is the utility agent i obtains for satisfying need j. $eTDur_l$ defines the expected travel duration to location l, $eEDur_k$ specifies the expected execution duration of activity k and $minEDur_k$ represents its minimal possible execution duration. Since *utility density* considers utility U, it respects personal preferences of agents. At the same time, it also incorporates travel durations which makes sure, that agents have a preference for locations which are close to their current location.

In a second step, the planning heuristic includes considerations about the flexibility to postpone activities. The end time of the previously determined activity-location pair defines the earliest time when the next activity can be executed. This time is important for the postponement because together with the expected travel duration and the expected execution duration it defines the end time of the next activity. This end point determines whether the next activity can be executed during the current or the next execution window. The heuristic value considering postponement is defined as (we omit indices for simplicity)

$$PP(w_1, w_2, \Delta t, N^t) = \begin{cases} w_1 \cdot N^{inc}(\Delta t, N^t) & \text{if } actE \leq cwe \\ w_2 \cdot N^{inc}(\Delta t, N^t) & \text{otherwise} \end{cases} \tag{6}$$

where $actE$ defines the end of the next activity, cwe represents the end of the current execution window, w_1 and w_2 are weight factors, Δt specifies the duration between the time when agent i satisfied need j for the last time and $actE$ and N^t represents the need level at the time when it was satisfied the last time (see Fig. 5 for a visualization).

The postponement function provides agents with an awareness of time pressure. This awareness drives agents to satisfy needs earlier even if it results in lower utility than the planned activity. This favors the satisfaction of needs with high need levels and thereby reduces the risk of missing the current execution window. At the same time, it also provides agents with an awareness of situations with a long waiting time until the next execution window starts (e.g. shops are closed during long weekends). This favors the satisfaction of needs whose need levels would grow to extreme levels during the necessary waiting time.

Based on the above calculations, the planning heuristic replaces the planned activity-location pair by the activity-location pair that yields the highest combined heuristic value. This value is defined as (we omit indices for simplicity)

$$HV(w_1, UD, PP) = w_1 \cdot UD + PP \tag{7}$$

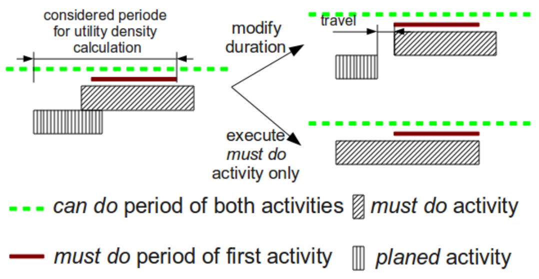
where w_1 defines a weight factor for the *utility density* value UD for the activity-location pair k, l and PP represents its postponement value.

At this point, the agent knows which activity-location pair it should execute as long as it does not violate a *must do* requirement of another activity-location pair (see section Execution Period for an explanation of *must do* requirements). Algorithm 2 and figure 6 outline the decision procedure agents use to recognize which activity-location pair they should execute. The selected activity-location pair is executed as planned as long as it does not violate the next *must do* execution period. Otherwise, the algorithm modifies the duration of the selected activity-location pair until it fits before the *must do* execution period. The algorithm executes both activity-location pairs if their combined *utility density* is higher than the *utility density* of the *must do* activity-location pair.

Algorithm 2 Decision procedure for the *must do* violation check

```
if $actE$ + $expMustDoTravelDur$ < $mustDoStart$ then
    execute selected pair
else
    shorten duration of selected pair to fit before must do pair
    $UD_{mustDo} \leftarrow UD(travel + mustdo)$
    $UD_{doBoth} \leftarrow UD(travel + selected + travel + mustdo)$
    if $UD_{doBoth}$ < $UD_{mustDo}$ then
        execute must do activity-location pair
    else
        execute modified selected pair and must do pair
    end if
end if
```

Figure 6: Visualization of the *must do* violation check


considered periode for utility density calculation
modify duration
travel
execute *must do* activity only
can do period of both activities
must do activity
must do period of first activity
planed activity

6 Calibration and Validation

The first part of this section illustrates how we adjust the behavior of agents based on the available calibration parameters. The aim is to provide an overview of the agent calibration mechanism and to show how one can engineer an agent with a desired behavior using these mechanisms. Thereafter, we demonstrate the behavioral patterns of fully calibrated agents. The last part of this section shows a performance analysis of our algorithms.

The agent we want to simulate in this example works full time and lives in a household with another full time working agent. Both agents have the same number of needs to satisfy which results in approximately the same workload. We simulate these agents over a period of 100 consecutive weeks and assign 13 unconstrained needs (sleep, work morning, work afternoon, daily housekeeping, house maintenance, garden maintenance, social contact, personal care, physical exercise, leisure green, leisure active, slack time and food) to each agent. In this example, we are going to include *grocery shopping* into the list of needs. We show how to adapt its configuration to illustrate how the behavior calibration mechanism works. The assumed need increase rate is two days.

- In a first simulation, we do not introduce any constraints to the need *grocery shopping*. Fig. 7(a) illustrates that agents satisfy the need at no particular time. In average, the need has a satisfaction rate of approximately 0.5 times per day as induced by the need increase rate of two days.
- In a second simulation, we restrict the execution period (opening hour) of the need from Monday to Saturday from 8.5am to 6.5pm. In contrast to the first simulation, agents restrict themselves to the predefined execution durations. Furthermore, they realize that shops are going to be closed during the night through the look ahead capability of the planning heuristic. As a consequence, more agents go shopping in the evening than in the morning. Additionally, almost 100 % of the agents refill their food stock during

Saturday. This is also a result of the look ahead capability which makes agents aware of the break during Sunday. Because the food stock lasts for approximately two days, almost all agents need to refill their stock again on Monday (see Fig. 7(b)).

- In a third simulation, we further restrict the execution period from Monday to Wednesday simulating a week with a long weekend. Agents realize the long weekend and refill their food stock during Wednesday. Since the need increase rate does not change during the long weekend, agents have a extremely high need to refill their food stock on Monday morning (see Fig. 7(c)).

- In a forth simulation, we reuse the configuration of the third simulation and change the need increase rate during the long weekend to four days. As a consequence, agents change their behavior on Monday and refill their food stock during the day (see Fig. 7(d)).

- In a fifth simulation, we reuse the configuration of the first simulation (no restrictions at all) and introduce a shopping appointment modeled through a *must do* execution period on Saturday between 2pm and 2.5pm. The need satisfaction pattern illustrated in Fig. 7(e) shows that 100 % of the agents go grocery shopping during that time which implies that agents keep their appointments.

- In a last simulation, we reuse the configuration of the first simulation (no restrictions at all) and share the need for *grocery shopping* between two household members. We introduce a *must do* execution period during bedtime where agents negotiate about the assignment of the shared need. Sharing a need between two agents is equivalent to dividing the responsibility of its satisfaction. Consequently, each individual agent has to satisfy the need only every second time resulting in a reduction of the work load. Accordingly, the average satisfaction rate of each individual agent is divided by two illustrated by the reduction from 0.35 to 0.18 (see Fig. 7(f) and Fig. 7(g)).

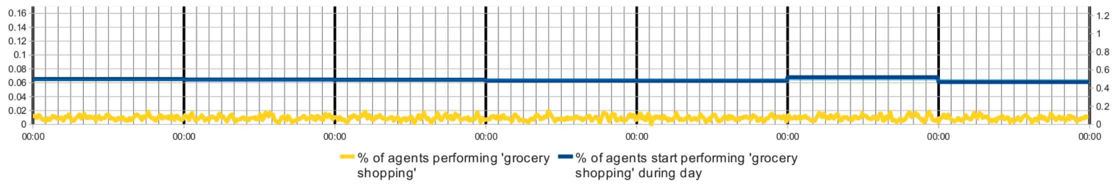
In a final step, we calibrate all considered needs (see Table 1) and run a simulation over a period of 100 equal weeks. Fig. 8 and Fig. 9 show the resulting behavioral patterns.

- During the week, agents primarily satisfy the need for *grocery shopping* after work. This pattern changes on Saturday since agents are less constraint (no work). Some agents decide to go *grocery shopping* during lunch break on Monday. This effect is a consequence of a high need level resulting from the impossibility to go *grocery shopping* during Sunday.

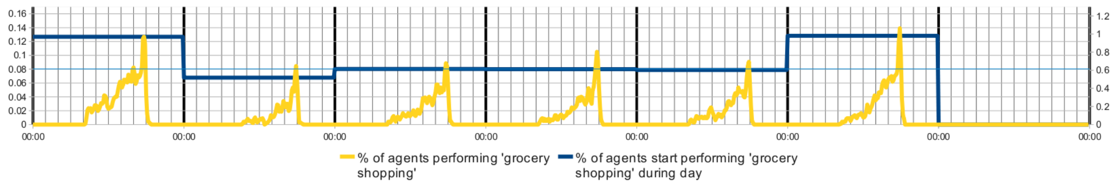
- *Daily housekeeping* is primarily satisfied during evenings. This pattern changes during the weekend since agents are less constraint (no work).

Figure 7: Illustration of the influence of calibration mechanisms to the behavior of agents. Samples are generated over a simulation period of 100 equal weeks. Please note that y-axes have different scales and that scales change in Fig. 7(e).

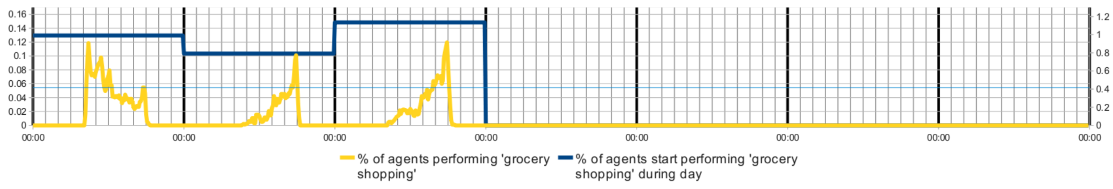
(a) Need satisfaction pattern with no restrictions.



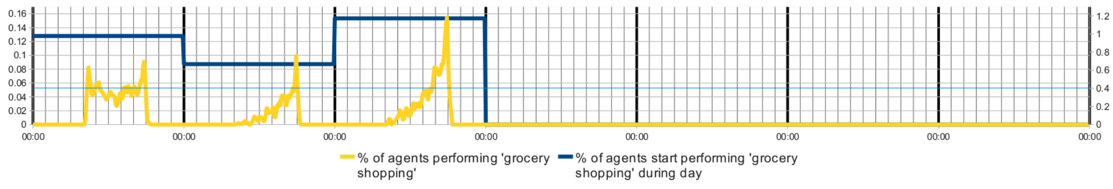
(b) Need satisfaction is restricted to Monday to Saturday from 8.5am to 6.5pm.



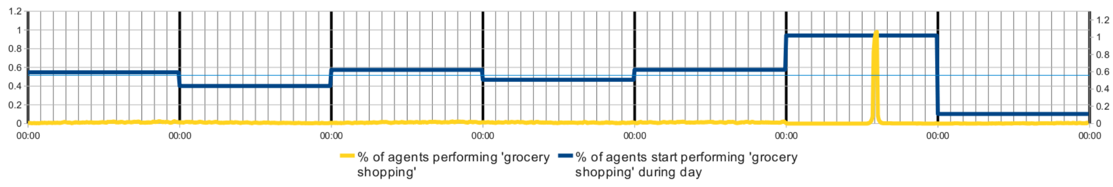
(c) Need satisfaction is restricted to Monday to Wednesday from 8.5am to 6.5pm.



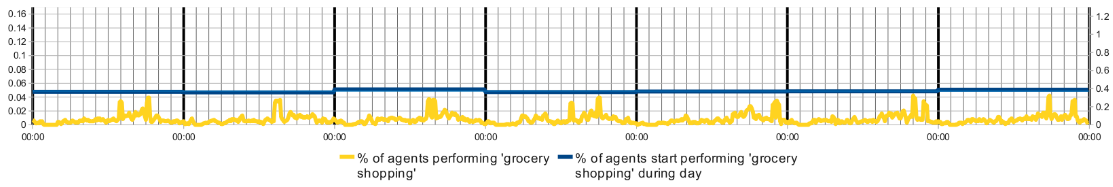
(d) Need satisfaction is restricted to Monday to Wednesday from 8.5am to 6.5pm with a different need increase rate from Thursday to Sunday.



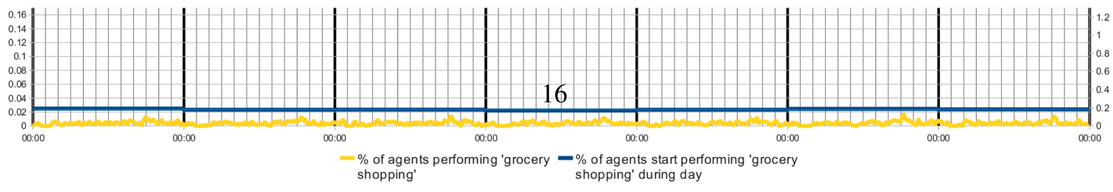
(e) Introduction of a shopping appointment on Saturday between 2pm and 2.5pm.



(f) Introduction of an appointment during bedtime where agents can negotiate about the assignment of the shared need (no shared need are defined yet).



(g) Introduction of an appointment during bedtime where agents negotiate about the assignment of the shared need *grocery shopping*.



- *Garden maintenance* and *house maintenance* show a similar pattern and are mainly performed during evenings and the weekend. *House maintenance* is not performed on Sunday because agents cannot obtain a reward at that day (due to a *cannot do* constraint).
- *Leisure active* and *leisure green* are equally configured and the configuration of *physical exercise* is also almost equal. Consequently, the behavioral pattern is also very similar. Agents preferably satisfy these two needs during the evening or lunch break. This pattern changes during the weekend where the satisfaction also takes place during the afternoon.
- *Personal care* is preferably performed in the morning or in the evening.
- *Work* is restricted to weekdays and there are periods where the presence of agents at the work place is required. Consequently, no agent works during the weekend and 100 % of the agents work during the week when their presence is required.
- The satisfaction of the need for *food* denotes a peak during lunch break. This peak is less remarkable during evenings and the weekend. Agents seem to have a intense workload because most of them do not have enough time to eat in the morning. This results in an average satisfaction rate of approximately twice during the week and increases to three times on Sunday.
- The need for *slack time* is primarily satisfied in the evening after work. This pattern changes during the weekend since agents are less constraint.
- The satisfaction of the need for *sleep* shows a similar pattern during weekdays. It changes on Friday and during the weekend since agents satisfy the need for *social contact* in the evening of those days.

6.1 Performance Measurement

The performance of the proposed simulation is important because we plan to use it for the simulation of very large scenarios. This is the reason why we design our algorithm for a distributed environment. At the same time, it is also important to use algorithms that scale well on the computation nodes. We use the configuration of the first simulation to do an empirical performance analysis of our code. This analysis shows that the code scales in $O(n)$ where n is the number of simulated agents. This is a satisfying result because it shows that the simulation time grows linear by the number of agents simulated by a computation node. The analysis also reveals that the code scales in $O(m \cdot log(m))$ where m is the number of needs handled by an agent. This is also a satisfying result, especially since individual agents will only be responsible for a small number of needs.

Figure 8: Behavioral patterns of all considered needs. Samples are generated from 1000 agents over a simulation period of 100 equal weeks. Please note that y-axes have different scales. This figure is continued in Fig. 9.

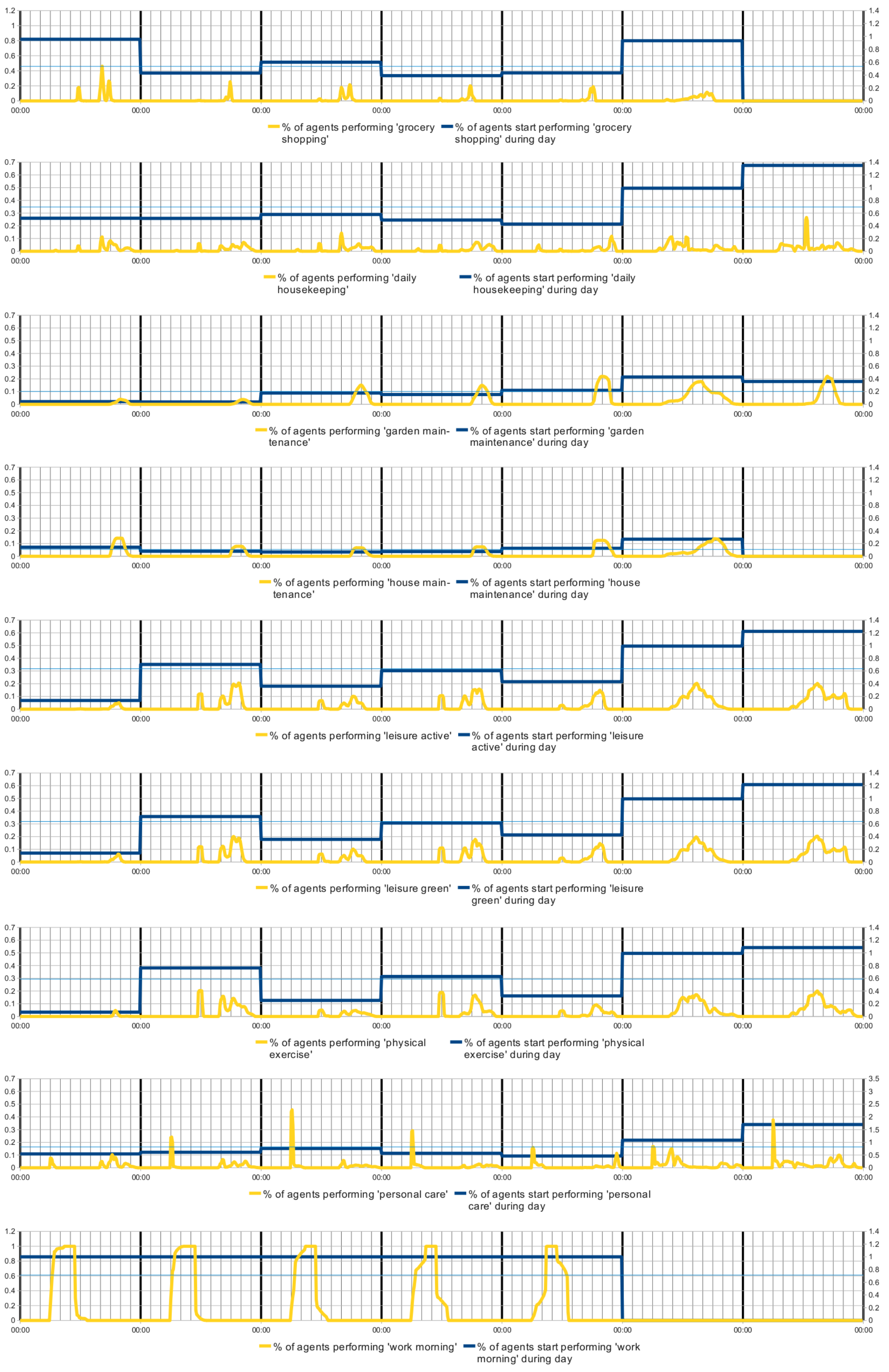


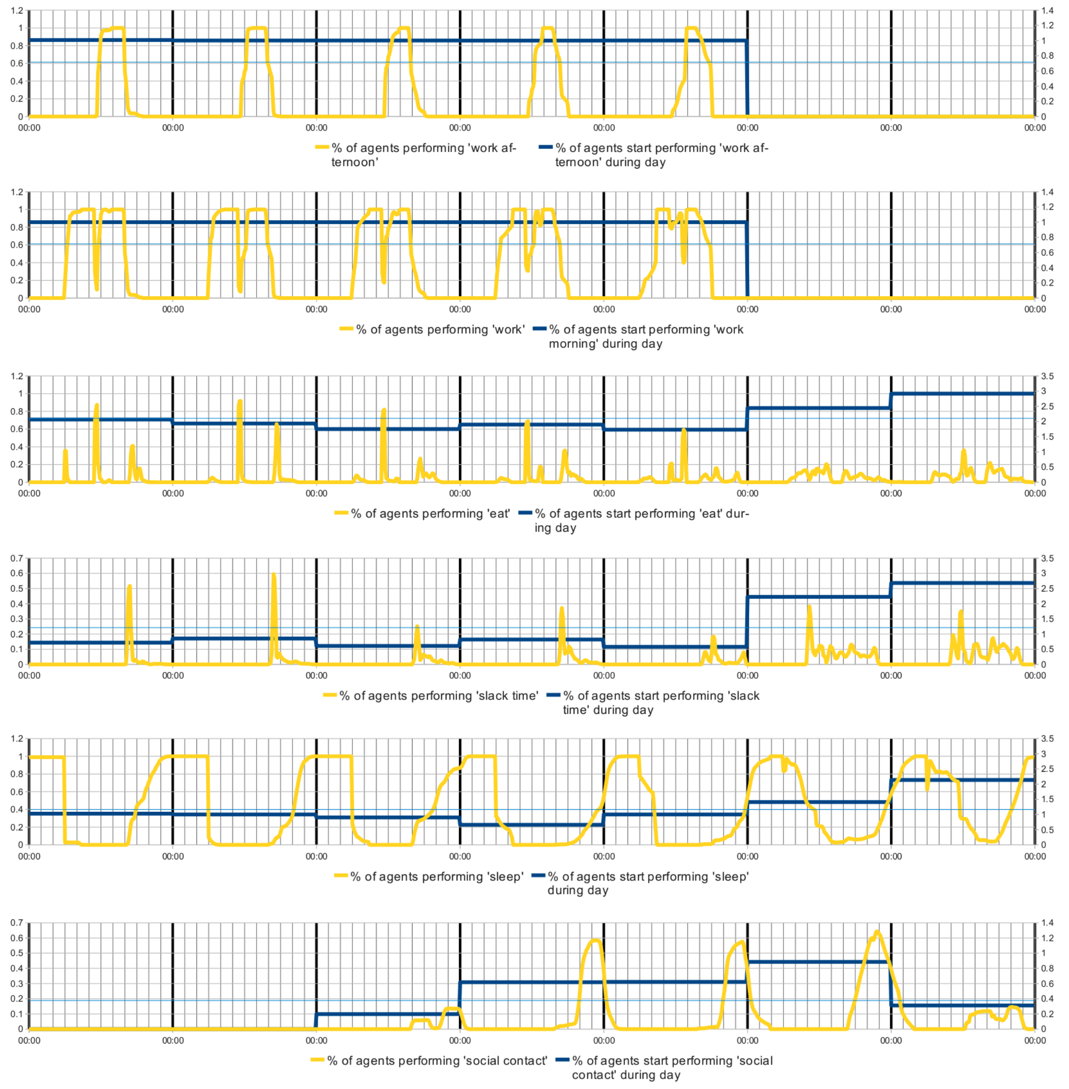
Table 1: Calibration of all considered needs of a working agent.

Need	Execution Period	Increase Duration	Decrease Duration
grocery shopping	Mo.-Sa.: 8.5am to 6.5pm (can do)	Mo.-Su.: 47.5h	Mo.-Su.: 0.5h
daily housekeeping	Mo.-Su.: 7am to 11pm (can do)	Mo.-Su.: 23.5h	Mo.-Su.: 0.5h
garden maintenance	Mo.-Fr.: 6pm to 10pm (can do) Sa.: 8am to 10pm (can do) Su.: 10am to 6pm (can do)	Mo.-Su.: 333h	Mo.-Su.: 3h
house maintenance	Mo.-Fr.: 6pm to 9pm (can do) Sa.: 8am to 9pm (can do)	Mo.-Su.: 332h	Mo.-Su.: 4h
leisure active	Mo.-Su.: 9.5am to 9pm (can do)	Mo.-Fr.: 47h Sa.-Su.: 23h	Mo.-Su.: 1h
leisure green	Mo.-Su.: 9.5am to 9pm (can do)	Mo.-Fr.: 47h Sa.-Su.: 23h	Mo.-Su.: 1h
physical exercise	Mo.-Su.: 9.5am to 11pm (can do)	Mo.-Fr.: 47h Sa.-Su.: 23h	Mo.-Su.: 1h
personal care	Mo.-Fr.: 6am to 11am (can do) Mo.-Fr.: 3pm to 12pm (can do) Sa.-Su.: 6am to 12pm (can do)	Mo.-Su.: 24h	Mo.-Su.: 0.5h
work morning	Mo.-Fr.: 6am to 1pm (can do) Mo.-Fr.: 9am to 11am (must do)	Mo.-Fr.: 19.75h Sa.-Su.: 48h	Mo.-Su.: 4.25h
work afternoon	Mo.-Fr.: 12am to 6pm (can do) Mo.-Fr.: 2pm to 3.5pm (must do)	Mo.-Fr.: 19.75h Sa.-Su.: 48h	Mo.-Su.: 4.25h
food	Mo.-Fr.: 5am to 9am (can do) Mo.-Fr.: 11am to 2pm (can do) Mo.-Do.: 17am to 9pm (can do) Fr: 17am to 12pm (can do) Sa.-Su.: 7am to 12pm (can do)	Mo.-Su.(night): 10h Mo.-Su.(day): 5h	Mo.-Su.: 0.5h
slack time	Mo.-Fr.: 4.5pm to 12pm (can do) Sa.-Su.: 10am to 10pm (can do)	Mo.-Fr.: 23.5h Sa.-Su.: 6h	Mo.-Su.: 0.5h
sleep	Mo.-Fr.: 10.5pm to 10am (can do) Sa.-Su.: 10.5pm to 1pm (can do) Mo.-Su.: 4am to 6am (must do)	Mo.-Su.: 15.5h	Mo.-Su.: 8.5h
social contact	Mo.-Fr.: 4pm to 4am (can do) Sa.: 5pm to 6am (can do) Su.: 11am to 10pm (can do)	Mo.-Fr.: 92h Sa.: 16h Su.: 92h	Mo.-Su.: 4h

7 Outlook

Due to the novelty of the presented simulation, it is important that we invest in validating this approach in more detail and thereby show its strengths and weaknesses. In section Calibration and Validation, we show that the model can be configured to produce various behavior. We plan to use these insights to engineer different agent types (e.g. working person, housewife etc.) and validate their simulated behavior with real world behavior. The validation could consist of following phases:

Figure 9: Fig. 8 continued.



1. Intra-day behavior validation based on weekdays. Thereby we plan to investigate following properties:
   - Numbers of activity types per weekday
   - Duration of activity types per weekday
   - Start time of activity types per weekday
2. Inter-day behavior validation. Thereby we plan to investigate following properties:
   - Average time intervals between activity types
   - Transition probabilities between activity types and weekdays

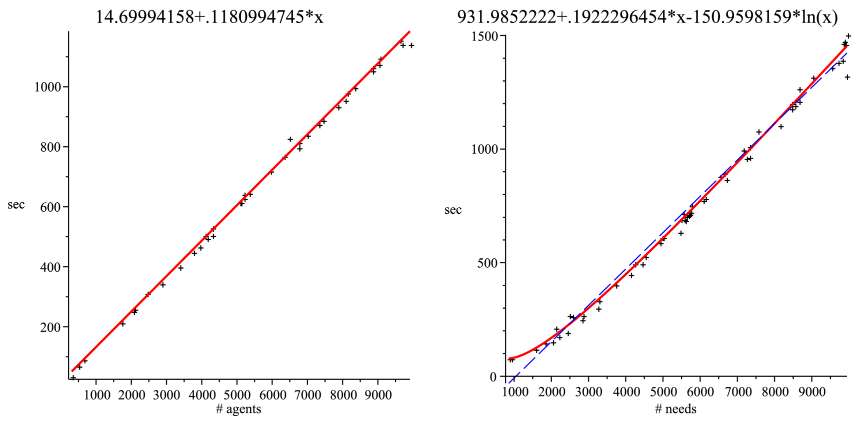


Figure 10: Empirical performance analysis of our code. Samples are generated over a simulation period of 100 equal days

(a) Analysis based on the number of agents (each agent manages 14 needs)

(b) Analysis based on the number of needs (simulation uses 10 agents)

3. Investigate the capability of our algorithm to produce diverse activity patterns
4. Similarities of weekdays based on the affinity-measurement of Joh (2004)

To be able to compare real life activity patterns with simulated ones, we need observed activity patterns of respondents showing their behavior over several weeks. We intend to use two existing six week continuous travel diaries (Zimmermann *et al.* (2001) and Löchl *et al.* (2005)) and analyses of activity patterns (e.g. Schönfelder (2006)) as the base for our validation.

We consider to implement an automated calibration and validation feature to further simplify the calibration process. It should provide users the possibility to specify a certain behavior. Thereafter, the user can run an algorithm which calibrates the defined agent type with the appropriate settings and validates its behavior.

We are thinking of using the concept of *projects* (e.g. Axhausen (1998), Miller (2005) and Schönfelder and Axhausen (2009)) as an extension of the need-based model. Axhausen defines a *project* as a coordinated set of activities, tied together by a common goal or outcome. Miller argues that *projects* are a reasonable organizing principle for dealing with complex human behavior. We see *projects* as a need generating mechanism which supplies agents' need-driven behavior with a continuous stream of needs. This could generate even more diversified behavior than the current approach which builds on recurrent needs. Bringing these two concepts together could also yield a pattern language where simple *projects* could serve as building blocks for more complex *projects* (similar to the ideas of Alexander *et al.* (1977) and Gamma *et al.* (1995)). Agents could reuse these *pattern projects* and implement them in their daily life.

This would lead to similar but still distinct behavioral patterns of individual agents.

8 Conclusion

This paper introduces a microscopic traffic simulation that simulates the behavior of agents and the resulting traffic continuously. The central component of the proposed behavioral model are needs and their development over time. We illustrate that it is possible to produce comprehensive behavior by combining the need-based approach with two calibration mechanisms. These calibration mechanisms are the adaptation of the need development over time based on the time of day and the day of the week and a mechanism that tells agents when they cannot, can or even must satisfy certain needs. We think that these are intuitive parameters which facilitate the calibration of the model. We designed our model for a distributed computation environment to accelerate the simulation. This imposes certain constraints to the model and the algorithms. We explain how the need-based model can be extended to support shared needs and still respect these constraints. Agents keep track of their need levels and provide them to a planning heuristic which makes *just in time* decisions about upcoming activities an agent should execute. The planning heuristic has a look ahead capability. This makes it possible to base decisions not only on the current need levels of an agent but also on the development of these levels in the near future. Making decisions *just in time* in a continuous manner enables agents to react to unexpected events and thus to overcome unrealistic behavior of the iterative optimization concept. The continuous nature of the simulation also helps to overcome performance issues and makes it possible to investigate traffic effects that occur between days and between weeks. We conclude by presenting how we use the calibration mechanisms to model distinct behavior. This demonstrates how agents can be engineered to show desired characteristics.

References

Alexander, C., S. Ishikawa and M. Silverstein (1977) *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, Oxford.

Arentze, T. A. and H. J. P. Timmermans (2006) A new theory of dynamic activity generation, paper presented at the *85th Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2006.

Arentze, T. A. and H. J. P. Timmermans (2009) A need-based model of multi-day, multi-person activity generation, *Transportation Research Part B: Methodological*, **43** (2) 251–265.

Atkinson, J. B. (1994) A greedy look-ahead heuristic for combinatorial optimization: An application to vehicle scheduling with time windows, *Journal of the Operational Research Society*, **45** (6) 673–684.

Axhausen, K. W. (1998) Can we ever obtain the data we would like to have?, in T. Gärling, T. Laitila and K. Westin (eds.) *Theoretical Foundations of Travel Choice Modeling*, 305–323, Pergamon, Oxford.

Balmer, M. (2007) Travel demand modeling for multi-agent traffic simulations: Algorithms and systems, Ph.D. Thesis, ETH Zurich, Zurich, May 2007.

Charypar, D., A. Horni and K. W. Axhausen (2009) Need-based activity planning in an agent-based environment, paper presented at the *12th International Conference on Travel Behaviour Research (IATBR)*, Jaipur, December 2009.

Charypar, D., A. Horni and K. W. Axhausen (2010) Pushing the limits: A concept of a parallel microsimulation framework, *Working Paper*, **640**, IVT, ETH Zurich, Zurich.

Charypar, D. and K. Nagel (2006) Q-learning for flexible learning of daily activity plans, *Transportation Research Record*, **1935**, 163–169.

Dantzig, G. B. (1957) Discrete-variable extremum problems, *Operations Research*, **5** (2) 266–288.

Feil, M., M. Balmer and K. W. Axhausen (2009) Generating comprehensive all-day schedules: Expanding activity-based travel demand modelling, paper presented at the *European Transport Conference*, Leeuwenhorst, October 2009.

Feil, M., M. Balmer and K. W. Axhausen (2010) New approaches to generating comprehensive all-day activity-travel schedules, paper presented at the *89th Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2010.

Gamma, E., R. Helm, R. Johnson and J. Vlissides (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston.

Ioannou, G., M. Kritikos and G. Prastacos (2001) A greedy look-ahead heuristic for the vehicle routing problem with time windows, *Journal of the Operational Research Society*, **52** (5) 523–537.

Joh, C.-H. (2004) Measuring and predicting adaptation in multidimensional activity-travel patterns, Ph.D. Thesis, Technical University Eindhoven, Eindhoven.

Kuhnimhof, T. and C. Gringmuth (2009) Multiday multiagent model of travel behavior with activity scheduling, *Transportation Research Record*, **2134**, 178–185.

Löchl, M., S. Schönfelder, R. Schlich, T. Buhl, P. Widmer and K. W. Axhausen (2005) *Untersuchung der Stabilität des Verkehrsverhaltens*, 1120, Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation, Berne.

Miller, E. J. (2005) An integrated framework for modelling short- and long-run household decision-making, in H. J. P. Timmermans (ed.) *Progress in Activity-Based Analysis*, 175–201, Elsevier, Oxford.

Nagel, K. and G. Flötteröd (2009) Agent-based traffic assignment: Going from trips to behavioral travelers, paper presented at the *12th International Conference on Travel Behaviour Research (IATBR)*, Jaipur, December 2009.

Richards, F. J. (1959) A flexible growth function for empirical use, *Journal of Experimental Botany*, **10** (2) 290–301.

Schlich, R. (2004) Verhaltenshomogene Gruppen in Längsschnitterhebungen, Ph.D. Thesis, ETH Zurich, Zurich.

Schönfelder, S. (2006) Urban rhythms: Modelling the rhythms of individual travel behaviour, Ph.D. Thesis, ETH Zurich, Zurich.

Schönfelder, S. and K. W. Axhausen (2009) Travel as a function of (life) projects, paper presented at the *European Transport Conference*, Leeuwenhorst, October 2009.

Simon, H. (1955) A behavioral model of rational choice, *Quarterly Journal of Economics*, **69**, 99–118.

Zimmermann, A., K. W. Axhausen, J. Beckmann, K. J. Beckmann, M. Düsterwald, E. Fraschini, T. Haupt, A. König, A. Kübel, G. Rindsfüser, R. Schlich, S. Schönfelder, A. Simma and T. Wehmeier (2001) Mobidrive: Dynamik und Routinen im Verkehrsverhalten: Pilotstude Rhythmik, *Final Report*, Federal Ministry of Education and Research, PTV and IVT, ETH Zurich and Institut für Stadtbauswesen (ISB), RWTH Aachen, Karlsruhe, Zurich and Aachen.