Accepted Manuscript

Integrating public transport into mobiTopp

Lars Briem, H. Sebastian Buck, Nicolai Mallig, Peter Vortisch, Ben Strasser, Dorothea Wagner, Tobias Zündorf

PII: S0167-739X(17)32007-1
DOI: https://doi.org/10.1016/j.future.2017.12.051
Reference: FUTURE 3889

To appear in: *Future Generation Computer Systems*

Received date : 7 September 2017
Revised date : 8 December 2017
Accepted date : 26 December 2017

Please cite this article as: L. Briem, H. Sebastian Buck, N. Mallig, P. Vortisch, B. Strasser, D. Wagner, T. Zündorf, Integrating public transport into mobiTopp, *Future Generation Computer Systems* (2017), https://doi.org/10.1016/j.future.2017.12.051

Integrating public transport into mobiTopp

Lars Briem[a,*], H. Sebastian Buck[a], Nicolai Mallig[a], Peter Vortisch[a], Ben Strasser[b], Dorothea Wagner[b], Tobias Zündorf[b]

[a] *Institute for Transport Studies, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*
[b] *Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*

Abstract

Demand and supply are both relevant for travel time in public transport. While it is obvious that the supply side in form of the timetable corresponds directly to the travel time, the demand side influences the travel time only partially, but in critical moments. During peak hours, when the demand reaches the capacity of the vehicles, the interaction between demand and supply becomes important. Overcrowded vehicles, hindering passengers to catch their chosen route, lead to longer travel times. Therefore, it is important to integrate the supply side of public transport into a travel demand model.

The supply side of public transport has been integrated into the agent-based travel demand model mobiTopp. A timetable has been implemented, which is used for two purposes. First, it serves as input for the Connection Scan Algorithm, which is used by the agents to find the routes with earliest arrival time at their destinations. Second, it is used for the movement of the public transport vehicles. The model also contains capacity constraints for vehicles, which, when activated, result in a noticeable increase in travel time.

An advanced version of the Connection Scan Algorithm allows creating travel time estimates between zones as matrices and profiles. Profiles, compared to matrices, contain for every departure time the corresponding arrival time, while matrices contain mean travel times. Both provide a flexible trade-off between realism and runtime.

Keywords: public transport; travel demand model; agent based; public transport assignment; route search; mobiTopp;

1. Introduction

Travel demand models are essential for the assessment of transport policy measures. They typically model the demand side of travel quite well. However, modelling only the demand side is not sufficient, as several of the networks' quality of service attributes, which enter as attributes into the demand model, like travel time, reliability, or comfort, depend on the interaction of the supply side and the demand side.

The supply side is typically only modelled for the motorized traffic, since for motorized traffic an increased demand results directly in an increased travel time. For public transport, travel time is at first independent from the demand and only depends on the supply side, i. e. the timetable. However, during peak hours, when capacity is reached, the situation changes.

An increasing demand firstly affects the comfort attribute, when there are no longer enough seats for all passengers. As the vehicle gets more and more crowded, the available space per passenger shrinks, creating more and more discomfort. With an increasing number of passengers standing in the aisles, entering and leaving the vehicle slows down, possibly creating additional delays of the vehicle and as such reducing reliability. In case of an overcrowded vehicle, it is even no longer possible for some public transport users to board. They have to wait for the next vehicle to arrive or even choose a new connection. So in this case, the demand affects the travel time for public transport.

Agent-based models offer a convenient way to integrate the supply side of public transport. The paper describes how the supply-side of public transport has been integrated into mobiTopp. The results show, how taking the capacity of public transport vehicles into account affects the resulting travel time.

2. Related work

In recent years, the supply side of public transport has been implemented in some agent-based models. One of the first implementations was done for MATSim in 2010. Rieser [1] implemented a basic model covering the timetable of one day with several lines, stops and vehicles. Similar work was done for SUMO [2] with the aim to integrate intermodality into SUMO by implementing public transport. POLARIS [3] currently does not have an implementation of the supply side of public transport. However, it is planned to model public transport for emergency scenarios, where it is necessary to guide drivers and optimize the routes to save as much lives as possible.

*Corresponding author. Tel.: +49-721-608-47772 ; fax: +49-721-608-46777.
Email address: lars.briem@kit.edu (Lars Briem)

Before agent-based models gained popularity in travel demand modelling, public transport has already been modelled in macroscopic models, for example in VISUM from PTV AG. VISUM contains a transit assignment procedure, which can be configured to take vehicle capacities into account. VISUM can also be seen as one of the larger public transport data models, covering many corner cases. As it is used by public transport companies to plan their operations, it also includes fare rules and costs for vehicles to calculate the overall costs for all operators [4]. This model and the model defined by the Association of German Transport Companies (VDV) [5] can be seen as industry standards for public transport data models.

Simulation experiments by Nuzzolo et al. [6] show that capacity constraints have a great influence on transit assignment. In these experiments, a large number of passengers change their departure time based on experience with overcrowded vehicles at earlier days.

Using detailed demand data during transit assignment implies to find routes for all agents. In the past, various algorithms have been developed to find routes in public transport networks. Schulz et al. [7] describe speed up techniques for Dijkstra's algorithm in public transport networks. The experiments are based on train data of a German railroad company and route search queries of a journey planner. Delling et al. [8] take a step further by introducing a new route search algorithm (RAPTOR), which is not Dijkstra-based. RAPTOR works in rounds. During each round RAPTOR processes all transit lines. For each transit line it updates the arrival times at all stops that are reachable without transfers. After n rounds, the current earliest arrival times represent the earliest arrival with at most $n-1$ transfers. Thus, RAPTOR automatically calculates Pareto-optimal routes considering arrival time and number of transfers. It works without preprocessing and is therefore able to handle dynamic scenarios, like search requests including real-time delays of vehicles. Besides this, RAPTOR is faster than previous Dijkstra-based approaches. Dibbelt et al. [9] introduce another transit route search algorithm based on a list of connections sorted by departure and arrival times to speed up the search requests further. They also introduce profile queries to calculate alternatives for the user within a single route search request. Profiles contain all earliest arrival times for each departure time within a given range.

There also exist approaches to combine transit assignment with profile based route search algorithms in an efficient manner. Grouping effects can be used to process a transit assignment quite fast [10]. Instead of an earliest arrival time as travel time, a perceived arrival time is used to assign agents to their routes.

3. The mobiTopp model

mobiTopp [11, 12] is a travel demand model implemented in Java. mobiTopp models every person, household and car of the planning area. Persons are modelled as individual agents, who are grouped together into households. An agent, according to the definition by Bonabeau [13], is an 'autonomous decision-making entity', which 'individually assesses its situation and makes decisions on the basis of a set of rules'.

mobiTopp is based on the principle of 'simulating activity chains' [14]: Each agent is assigned an activity program for a whole week, which he executes during the simulation. Executing the activity program, the agent chooses locations for his activities and modes for the trips necessary to reach these locations. The decision rules for these choice situations, destination choice and mode choice, are implemented as separate classes implementing a specific interface. Different implementations exist, which are currently all based on Discrete Choice models, namely Logit and Nested Logit models [15]. These models take sociodemographic attributes of the agents and network variables like travel time and (monetary) travel cost into account. Travel time and travel cost on a zone-to-zone basis is calculated externally and provided in the form of matrices as input to mobiTopp. Travel time can be differentiated by time of day with time slices of multiples of an hour. So peak-hours and off-peak periods can be distinguished.

mobiTopp currently supports the modes *cycling*, *car as a driver*, *car as a passenger*, *walking*, and *public transport* as default, additional modes like *carsharing* can be provided via extensions [16]. Traffic assignment is currently handled externally using a macroscopic traffic assignment tool. For this purpose mobiTopp's output is aggregated to origin-destination matrices, either over the whole day or hourly, depending on the needed granularity for the macroscopic traffic assignment. Due to this , individual vehicles, like cars and bikes, do not interact with each other directly. Agents only interact with the cars of their household by reserving the car while using it. Reserving cars is handled in a first-come-first-served manner. The first agent needing a car takes it. Reserved cars are not available to other agents of the household for the mode *car as a driver* until the car is returned at home.

mobiTopp consists of two stages: initialisation and simulation. Initialisation comprises population synthesis and setting up the whole system considering long-term aspects and decisions of persons and households. During this stage, agents are assigned fixed locations for home, work and education activities. Agents are also equipped with private cars and transit passes.

During the simulation stage, the travel behaviour of agents is modelled, consisting of performing activities, mode choice, destination choice, and making trips. All agents are simulated simultaneously over the period of one week. The behaviour of agents is encapsulated into states. Starting at home, agents start in the state *execute activity*. An agent remains in this state until the end of the activity. He then switches to the state *make trip*. While switching, the agent makes decisions for the next location and mode to take. After the trip, the agent switches back to *execute activity*. As soon as there are no more activities to pro-

ceed, the agent stays at home in *finished* state for the rest of the simulation. Besides making decisions during state transitions, agents can execute actions on state changes, too. E. g. after a trip in mode *car as driver* is finished and the agent arrives at home, the agent has to return the car. It is now possible for other members of the household to use the car.

States as described here have different types. The states *execute activity* and *make trip* are so-called duration states. An agent remains inside these states for a longer time. All other states are instantaneous states. In these states, the agent executes a specified behaviour and switches to the next state without consuming time. All states are described in Figure 1.

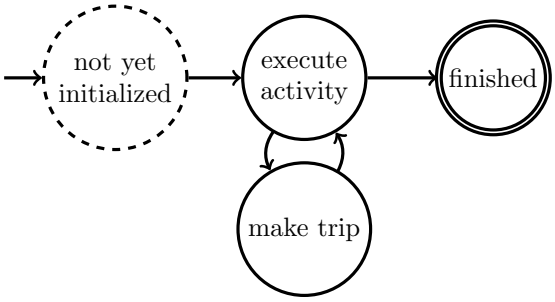

Figure 1: State diagram of an agent in mobiTopp [17]. Duration states are marked with solid lines and instantaneous states are marked with dashed lines. During the simulation, the agent alternates between the states *execute activity* and *make trip* [18].

4. Public transport in mobiTopp

The behaviour of persons in public transport is quite different from travelling by other modes like car or bike. In public transport networks, persons can only board or exit a vehicle at specific stations. Therefore, they need the possibility to find a route from their current location to their destination, maybe using several vehicles on fixed routes instead of only one vehicle. Instead of developing a new algorithm, we decided to choose one that fits our needs. As the routing algorithm will be used heavily during simulation, the speed of a single route search request is quite important. The Connection Scan Algorithm [9] is one of the fastest algorithms currently available [19] and is hence selected.

4.1. Data model

The integration of public transport into the simulation requires a timetable, which provides every information needed by the route search algorithm. In addition, this timetable can be used for scheduling vehicles over the whole simulation period.

The definition of the data structures for the timetable starts with a minimal model, which is extended as necessary. The simplest model to find routes in a network consists of *stops* and *connections*. A stop is a location where agents can wait for, board, or exit a vehicle. A connection is a direct link between two stops with a given departure and arrival time. When transferring this into graph theory, stops correspond to nodes and connections to edges. Based on this, every graph-based algorithm, which is able to deal with time dependent edges, can be used for routing.

To be able to take transfer times into account during route search, the algorithm also needs information when transfer times have to be considered. Transfer times are needed when a passenger needs to change the vehicle. Therefore, all connections served by the same vehicle are grouped into a (vehicle) journey. A journey is the sequence of connections served by a vehicle on its trip from its origin to its final destination. Here, we combine the modelling of a vehicle and a journey and add the capacity of the vehicle directly to the journey.

Using only connections and stops is not enough to find routes. Situations exist where agents have to transfer from one stop to another. Therefore, transfers are modelled as footpaths in the neighbourhood of each stop. A neighbourhood contains all stops that are reachable from a stop within a given period of time, e.g. 15 minutes. Neighbourhoods are explicitly not modelled as stations, since there can be transfers between different stations. Stations, as aggregates of stops, are also modelled, primarily to make analysis easier.

In addition to the components needed for routing, the model contains other elements for easier conversion from other models and for visualisation. These are journey templates, transport systems and route points. Journey templates describe departure and arrival times at stops. Times are given relative to the first stop. The departure time at the first stop is always zero. Combining a journey template with an absolute departure time, including a date, results in a journey and the corresponding connections. Route points represent the geographic route of a journey. They are currently only needed for visualisation. A complete overview of the model is given in Figure 2.

4.2. Route search

The Connection Scan Algorithm (CSA) is a fast transit route search algorithm linear in the number of connections. It takes as input the origin stop s_o, the destination stop s_d, the departure time t_{dep}, all connections C, forming the timetable, and the footpaths F between each stop and the stops in its neighbourhood. A connection $c = (c_{from}, c_{to}, c_{dep}, c_{arr}, c_{journey})$ belongs to a journey $c_{journey}$ and represents the basic movement of a vehicle from a stop c_{from} at departure time c_{dep} to the following stop c_{to} with arrival time c_{arr}.

The algorithm itself consists of an initialisation and the linear sweep over all connections. During the initialisation all earliest arrival times τ are initialised with infinity, except the earliest arrival times at the origin stop s_o and its neighbours. The list of connections is sorted by increasing departure time c_{dep} and, as secondary sorting criterium, increasing arrival time c_{arr}.

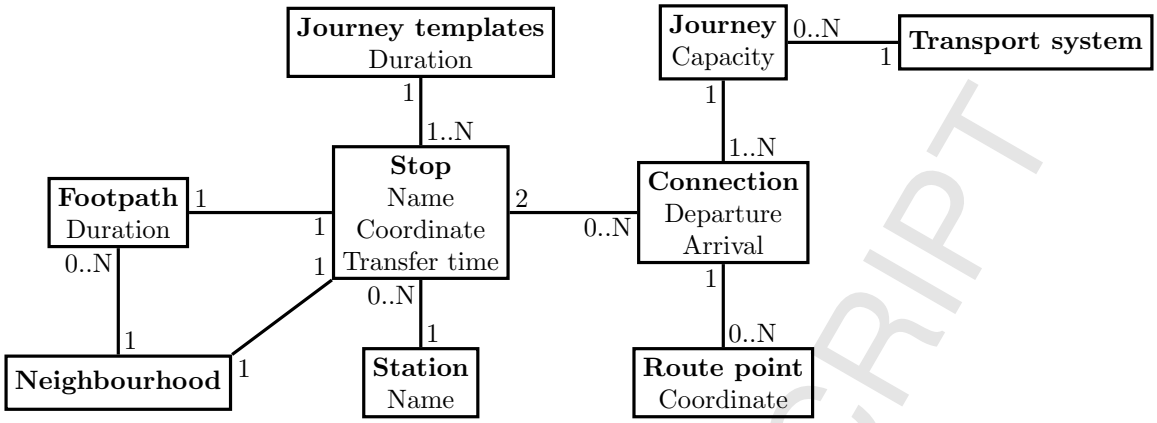

Figure 2: Complete overview of the timetable in mobiTopp [18]

The second part of the algorithm is a linear sweep over all connections, updating earliest arrival times τ at stops. An arrival time is updated if a connection is reachable and the arrival time of the current connection improves the earliest arrival at the stop.

Reachable means that the earliest arrival time at the origin stop of the connection $\tau(c_{from})$ is earlier or equal to the departure time c_{dep}, considering necessary change times τ_{change_time}. Change times are only relevant, if the journey of the connection $c_{journey}$ has not already been used.[1] A connection improves the earliest arrival time at the destination stop c_{to} if the earliest arrival time at the destination stop of the connection $\tau(c_{to})$ is later than the arrival time of the connection c_{arr}. If the arrival time is updated, all footpaths of this stop are used to check and possibly update the earliest arrival time of all stops in its neighbourhood [9, 20]. An update of the earliest arrival time via footpath does not trigger an update of other arrival times. This prevents routes purely made up of footpaths. The algorithm is described in pseudocode in Listing 1.

Dibbelt et al. [9] and Strasser and Wagner [20] explain several techniques, which can reduce the runtime in real applications. One of those is, limiting the number of processed connections. Connections departing before the departure time of the search request can be ignored. The scan can also be stopped as soon as the next processed connection departs after the earliest arrival at the destination $c_{dep} > \tau(s_d)$ since no connection can reduce the earliest arrival time any further. Listing 1 does not contain such optimizations to present the algorithm as clear as possible.

[1] In the original CSA publication [9] the term *trip* has been used to denote the sequence of connections served by the same vehicle during its journey from the first stop to the last stop. As trip typically refers to person trips in travel demand modelling, the term *journey* will here be used where the original CSA publication uses *trip* (see Figure 2).

4.3. Behaviour

The original behaviour of agents in mobiTopp, as shown in Figure 1, is no longer sufficient for the detailed modelling of public transport. Therefore, the *make trip* state has been replaced with a more detailed behaviour, in case the agent uses public transport. After finishing an activity, instead of switching to *make trip* an agent switches to a system of states, which starts with the state *use public transport* and finishes with the state *leave public transport* (see Figure 3). Both states are used for analysis purpose.

On the transition from *use public transport* to *find route*, the agent searches footpaths to the next stops near him. In *find route* the agent uses those stops to search for his next route and walks to the stop, where his first vehicle departs, the origin of his route. Currently, the agent always uses the route with the earliest arrival time at his destination. After arriving at the origin stop, the agent *waits for the vehicle* he wants to board. If it is available, the agent *tries to board* it. While trying to board, the vehicle accepts or rejects the boarding attempt, based on the available remaining capacity. If the agent is allowed to board the vehicle, he starts to ride the vehicle (*ride vehicle*). Contrarily, if the agent is not allowed to board the vehicle, he has to find a new route, which starts at his current stop or one of its neighbouring stops. After a potential walk to the origin of the new route, the agent waits for the first vehicle serving the new route. As overcrowded vehicles typically appear only during rush hour, there should be no problem for agents to find another route, possibly the next vehicle of the same line.

The agent stays in the vehicle until he arrives at his destination stop or the next transfer stop. In both cases, the agent leaves the vehicle. In case he reaches his destination stop, the agent walks to his destination and *leaves public transport*. If not, the agent searches for the next vehicle or walks to the stop where his next vehicle departs. *Wait for vehicle* and *ride vehicle* are modelled as duration states, like *make trip* in the original model.

Algorithm 1 Connection Scan Algorithm, considering footpaths between stops, based on Dibbelt et al. [9].

function CSA(origin s_o, destination s_d, departure t_{dep}, connections C, footpaths F)
  $\tau(\cdot) \leftarrow \infty$ ▷ Initialise earliest arrival at all stops
  $\tau(s_o) \leftarrow t_{dep}$ ▷ Initialise arrival at origin
  for all $(s_o, s') \in F$ **do** ▷ Process all footpaths to neighbours of origin
    $\tau(s') \leftarrow t_{dep} + \tau_{footpath}(s_o, s')$ ▷ Initialise earliest arrival at neighbours of origin
  end for
  $J \leftarrow \emptyset$ ▷ Initialise used journeys
  sort C ▷ Sort connections ascending by departure and arrival time

  for all $c \in C$ **do**
    if $c_{dep} \geq \tau(c_{from}) + \tau_{change\_time}(c_{from})$ **or** $c_{journey} \in J$ **then** ▷ If connection is reachable or journey has been used
      if $c_{arr} < \tau(c_{to})$ **then** ▷ and arrival time is earlier
        $\tau(c_{to}) \leftarrow c_{arr}$ ▷ update earliest arrival time
        add $c_{journey}$ to J ▷ Add journey of connection to used journeys
        for all $(c_{to}, s') \in F$ **do** ▷ Relax all neighbours of current stop
          if $c_{arr} + \tau_{footpath}(c_{to}, s') < \tau(s')$ **then**
            $\tau(s') \leftarrow c_{arr} + \tau_{footpath}(c_{to}, s')$ ▷ Update earliest arrival time at neighbour
          end if
        end for
      end if
    end if
  end for
  return $(s_o, s_d, t_{dep}, \tau(s_d))$
end function

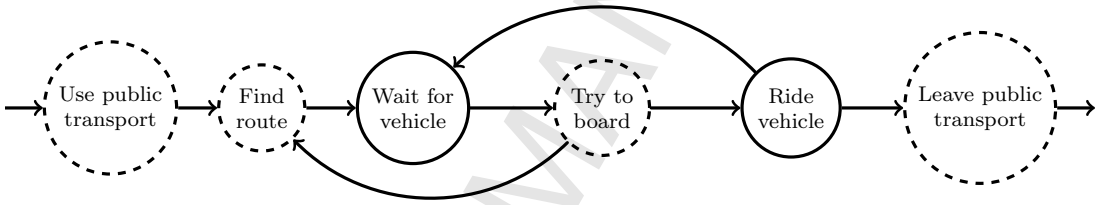

Figure 3: Visualisation of state changes for a person using public transport. *Wait for vehicle* and *ride vehicle* are duration states (solid border) in which an agent stays for a longer time. The other states are instantaneous (dashed border) and are used to encapsulate specific behaviour [18].

5. Results for transit assignment

The implementation of public transport in mobiTopp, which has been calibrated based on the data of a recent household travel survey [21], has been successfully applied to the region of Stuttgart. The region contains the city of Stuttgart and the surrounding districts. The model of the region consists of 1174 zones. All 2.7 million inhabitants of the study area are simulated over a period of one week.

The input data used to construct the timetable has been taken from a given macroscopic model [22]. The resulting timetable contains 785 118 connections, 48 100 journeys, and 24 different vehicle types, which serve 13 941 stops. The stops are combined to 11 410 stations. Between stops, 48 236 bidirectional footpaths exist. The footpaths specified in the macroscopic model have been extended by a footpath search to find all stops within 15 minutes walk time around each stop.

Using the presented model in mobiTopp results in about 1 million trips made by almost 620 000 agents. One simulation took about 48 hours to simulate a single day of travel demand on an Intel Core i7 3820 (3.6 GHz) using 24 GB of RAM. A mobiTopp run without public transport assignment takes for the same simulation period about 3 hours. The increased runtime is mainly the result of the large number of route search requests, including footpath search to and from stops.

The new public transport implementation allows a more realistic representation of public transport trips, heading from one activity location to another. Figure 4 shows the distribution of travel times in public transport, with and without capacity constraints. Travel times are distributed over a wide range, but a huge number of trips take less than one hour and most of the trips are faster than one hour and a half. The results show that the output of the simulation run with the constrained capacity contains fewer trips with shorter duration and more trips with longer duration than the output of the simulation run with unconstrained capacity. Constraining the capacity increases the mean travel time from 51 minutes to 1 hour and 7 minutes.

The distribution over the day of the number of passengers in public transport in Figure 5 contains three peaks. The morning peak, which is the highest one, is nearly double the size of the peak short after noon, while the peak in the evening lies in between both peaks. The morning peak is higher than the other peaks since school and work

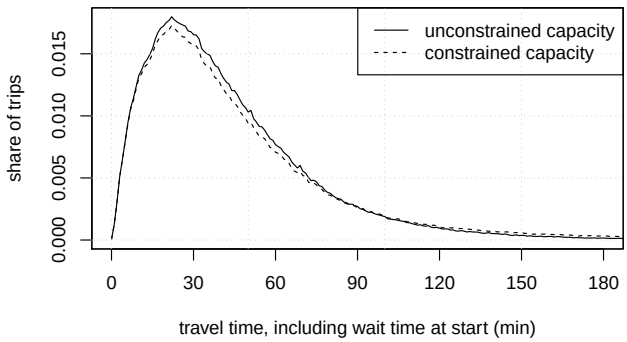

Figure 4: Distribution of travel times in public transport in mobiTopp, with and without capacity constraints. The travel time is measured from the arrival of an agent at the first stop to the exit at the last stop and includes waiting time at transfers.

start roughly at the same time in the morning, while the return trips are split into the noon peak and the evening peak.

The number of passengers in vehicles follows the curve of public transport users quite similar, but at a lower amplitude. Both curves start at a low level and decline after all peaks at the end of the day. In contrast, the number of waiting passengers starts also at a low level, but remains constant high at the end of the day, while the evening peak is not as noticeable as on the others. The number of waiting passengers is quite high compared to the number of passengers in vehicles.

We assume that this is mainly an effect of the activity plans. Activity plans are generated for agents with fixed activity durations. Meaning, that as soon as the activity has been started, the end time of an activity is fixed. As a trip starts, as soon as an activity is finished, the time to start the next trip is fixed, too. So the agent walks to the next stop, as soon as his activity is finished without considering the departure time of his connection. Therefore, he possibly has a considerable waiting time at the initial stop, which results in a high number of waiting passengers. In reality, however, it can be assumed that people are aware of the timetable and adjust the start time of their walk to the transit stop with the aim to minimize their waiting time. Thus, the behaviour of mobiTopp's agents should be adjusted to be more realistic in this aspect. This could be handled by some flexibility in the durations of the activities.

The effect of the waiting time at the initial transit stop is especially important for agents taking low frequency lines, because the additional wait time increases the travel time of the agent. Thus, the next activity and also the next trip starts later. In the evening, when the frequencies of lines start to decline, this could result in even longer wait times and therefore more agents waiting at stops. Which could be the cause for the high number of waiting passengers shown in Figure 5. Thus, the detailed implementation of public transport clearly shows the need for and leads the way to further improvements of mobiTopp regarding more flexible activity plans.

Besides the changes in the travel time distribution, enabling the capacity constraints, has another effect. Persons rejected to board a vehicle increase the number of persons waiting. However, the effect is small (see Figure 5). As the maximum number of rejected passengers is only 1.4% of the maximum number of agents in public transport, the curve is shown in more detail in Figure 6. The highest peak is during the morning rush hour, where we expect the most crowded vehicles. During the rest of the day, there are still some passengers being rejected, but not that many. In the evening, the number of rejected passengers starts to rise again. As there are also too many passengers waiting in the evening, we assume that both effects are interrelated. There is no obvious cause-effect relationship between passenger waiting and passengers not allowed to enter a vehicle, since one implies the other. We assume therefore that both effects have a common cause, possibly an overestimation of demand for public transport in the evening.

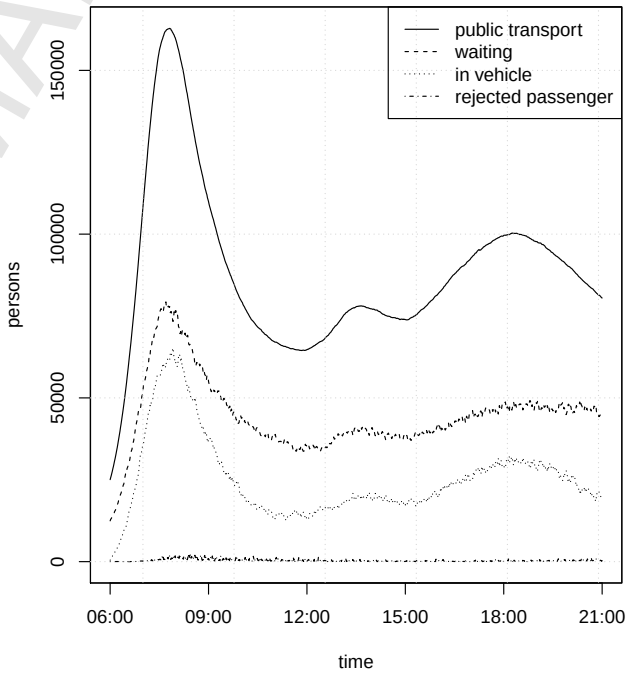

Figure 5: Distribution of persons per minute in public transport, waiting for a vehicle, riding a vehicle and being rejected to enter a vehicle.

6. Detailed travel times in mode and destination choice

Integrating detailed public transport makes the model more realistic, but also drastically increases runtime during route choice. Using detailed route search for the calculation of travel times in destination and mode choice will

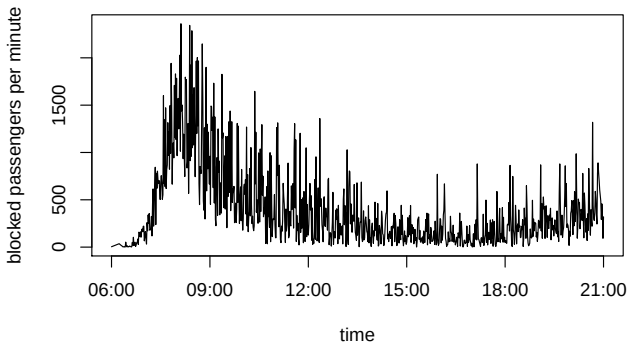

Figure 6: Distribution of passengers per minute being rejected to enter a vehicle.

further increase runtime, because in destination choice, the travel time to many destinations is needed. Using single route search requests will result in a huge overhead. As there are a lot of duplicate route search requests between stops, requests can either be cached or precalculated to reduce the runtime. The CSA in its basic form without performance extensions, explicitly without cancelling to sweep connections, calculates all earliest arrival times to all stops. These values could be cached and reused for all agents starting at the same stop until the next vehicle departs. Another option consists of introducing a preprocessing step before the whole simulation. The arrival times can be stored on disk and reloaded on later runs for other populations. Thus, a recalculation is only needed on timetable changes. Therefore, preprocessing is used instead of caching.

Preprocessing is especially a viable option if earliest arrival times can be calculated for several departure times in parallel. An extension of the CSA algorithm, the profile CSA Dibbelt et al. [9], allows this in a single sweep over all connections. Instead of calculating a single arrival time per stop, the profile CSA calculates all earliest arrival times for a given range of departure times, a so-called profile.

A profile is a function that maps each departure time to the earliest arrival time. Typically, this is calculated for a given origin stop, destination stop and a period of the departure time. A profile can also be calculated for the whole period of the timetable, representing for each departure in the timetable the earliest arrival time between the two stops. As the CSA works in a one-to-all fashion, it is possible to calculate profiles for a given origin stop to all destination stops during a single sweep over all connections.

While the profiles remain exact in time and location, the memory footprint is quite high. The required memory mainly depends on the number of connections in the timetable. For each connection an entry in the profile might be added. An entry consists of a departure and an arrival time, each represented by a 4 Byte integer. As the timetable used contains about 785.000 connections, which results in approximately 6 MB per profile. Using one profile for each of the 13941 stops will therefore result in 81 GB of data. Taking footpaths into account, a connection can be responsible for more than one profile entry. However, experiments have shown, that a profile including footpaths in our case needs between 4 MB and 5 MB, which is slightly lower than the theoretical size.

The memory footprint can be further reduced, if it is acceptable to ignore the exact location of an agent within a zone and use only the information of the zone instead. Then, travel times have to be calculated only between zones. In order to calculate travel times between zones a new stop is created for each zone's centroid, called zone stop. The zone stop is added to the neighbourhood of each stop in the zone and vice versa. The time to walk between the zone stop and the other stops can either be zero or the access time from the zone's centroid to the stop. The resulting travel time between zones already include the footpath to the stop where the agent can enter public transport.

The memory footprint of a single profile to a zone stop remains the same, because the limiting factor is the number of connections. Instead of one profile for each of the 13941 stops, only one profile per zone is calculated. Given the 1174 zones and one profile per zone, this results in approximately 6.87 GB of data. Thus, the profiles to zone stops reduce the memory footprint to only 8.4% compared to profiles to all stops.

Since profiles as data structure are more complex than matrices, we expect that the gain in detail, when using profiles instead of matrices in mode and destination choice, comes at the cost of increased runtime. In order to isolate the increase of runtime due to use of profiles in mode and destination choice, from the increase in runtime due to the detailed transit assignment implementation, we disabled the detailed transit assignment implementation for this comparison. The results show that by using profiles instead of travel time matrices, the runtime increases by a factor of two. This seems acceptable, especially compared with the increase in runtime by the detailed transit assignment.

Using the zone profiles, the memory usage is now manageable. However, there are still cases were memory or runtime limitations still do not allow to use exact departure times or where exact times are not needed. In this case, the original implementation based on travel time matrices can still be used. The matrices can either be provided externally as input or calculated based on profiles.

Forming travel time matrices based on profiles means, aggregating the travel times of all routes between zones in a given time slice. Therefore, one day is split up into 24 matrices, one for each hour. The estimated travel time for each slice will be a mean of the routes between zones. The granularity of one hour is sufficient to cover differences between peak hours and the rest of the day. As a

result, travel time matrices no longer have to be calculated externally. The memory consumption of a single matrix is fixed, because for each zone combination there is one matrix entry. As 1174 zones are used and travel times are stored using floating point numbers of 4 byte length, the memory consumption of a single matrix is 5.2 MB. Using 24 matrices will thus result in about 126 MB for all matrices.

7. Conclusion and future work

We have shown how the supply side of public transport can be integrated into an agent-based travel demand model. The current implementation allows already analysing the effect of vehicle capacities on the simulation results and hence allows assessing the potential bias of models neglecting vehicle capacities.

The presented approach follows the modular fashion and basic principle of mobiTopp, using simple models where possible, but providing the possibility to replace them with more sophisticated models when needed. In this case, such a simple model is the route choice behaviour. Currently, the route with the earliest arrival is chosen. An improved implementation could present a set of alternatives to the agent, differing for example by arrival time, travel time, and number of transfers. The agent could then choose from this set of alternatives, using a Discrete Choice model. Looking further, we can also integrate the agents experience about empty or crowded vehicles into his decision.

The modelling of precise departure times, instead of departure time periods, stimulates the improvement of other aspects of mobiTopp. One such aspect is flexibility of activities in terms of start time and duration. In reality, at least some persons using public transport can be expected to choose their activity times appropriate to departure times of public transport vehicles. Thus, waiting times of such persons can be reduced. In contrast, other persons are not able to chose their times freely, because of fixed working times.

The runtime of the current public transport implementation is a multiple of the runtime without it, mostly as a result of the route search and the interaction between agents and vehicles. Therefore, it is necessary to increase the performance of the route search significantly. An improvement is using approximations where accuracy is not indispensable. Depending on the runtime requirements either approximations of only location (zone profiles) or also time (matrices) can be used. mobiTopp is capable of handling both and thus provides a flexible trade-off between realism and runtime.

The presented work, integrating the supply side of public transport into mobiTopp, including the capacity constraints on single vehicles, provides a solid foundation for further extensions. The resulting implementation provides the potential to account for discomfort in the mode choice model, using the occupancy rate, and to integrate delays resulting from prolonged boarding times due to overcrowded vehicles. For these use cases, additional empirical data is needed.

Acknowledgement

This paper is an extended version of a paper [18] presented at the *6th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS 2017)*.

The work has been supported by grant FOR 2083 from Deutsche Forschungsgemeinschaft (DFG).

References

[1] M. Rieser, Adding Transit to an Agent-Based Transportation Simulation: Concepts and Implementation, Ph.D. thesis, Technischen Universität Berlin, Berlin, 2010.

[2] M. Behrisch, J. Erdmann, D. Krajzewicz, Adding intermodality to the microscopic simulation package SUMO, in: The Middle Eastern Simulation and Modelling Conference, 2010.

[3] V. Sokolov, J. Auld, H. Ley, M. Boltonc, Coordinated Transit Response Planning and Operations Support Tools for Mitigating Impacts of All-Hazard Emergency Events, in: World Conference on Transport Research, Shanghai, 2016.

[4] PTV AG, PTV Visum 15 – Manual, PTV AG, 2015.

[5] Verband Deutscher Verkehrsunternehmen (VDV), VDV-Standardschnittstelle Liniennetz/Fahrplan, 2013.

[6] A. Nuzzolo, U. Crisalli, L. Rosati, A schedule-based assignment model with explicit capacity constraints for congested transit networks, Transportation Research Part C: Emerging Technologies 20 (1) (2012) 16–33.

[7] F. Schulz, D. Wagner, K. Weihe, Dijkstra's algorithm on-line: an empirical case study from public railroad transport, Journal of Experimental Algorithmics (JEA) 5 (2000) 12, ISSN 1084-6654.

[8] D. Delling, T. Pajor, R. F. Werneck, Round-Based Public Transit Routing, Transportation Science 49 (3) (2015) 591–604, ISSN 0041-1655.

[9] J. Dibbelt, T. Pajor, B. Strasser, D. Wagner, Intriguingly simple and fast transit routing, in: International Symposium on Experimental Algorithms, Springer, ISBN 978-3-642-38527-8, 43–54, 2013.

[10] L. Briem, S. Buck, H. Ebhart, N. Mallig, B. Strasser, P. Vortisch, D. Wagner, T. Zündorf, Efficient Traffic Assignment for Public Transit Networks, in: C. S. Iliopoulos, S. P. Pissis, S. J. Puglisi, R. Raman (Eds.), 16th International Symposium on Experimental Algorithms (SEA 2017), vol. 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, ISBN 978-3-95977-036-1, ISSN 1868-8969, 20:1–20:14, 2017.

[11] N. Mallig, M. Kagerbauer, P. Vortisch, mobiTopp – A Modular Agent-based Travel Demand Modelling Framework, Procedia Computer Science 19 (2013) 854–859, ISSN 18770509.

[12] N. Mallig, P. Vortisch, Modeling travel demand over a period of one week: The mobiTopp model, arXiv preprint arXiv:1707.05050 .

[13] E. Bonabeau, Agent-based modeling: Methods and techniques for simulating human systems, Proceedings of the National Academy of Sciences (PNAS) 99 (suppl. 3) (2002) 7280–7287.

[14] K. W. Axhausen, R. Herz, Simulating activity chains: German approach, Journal of Transportation Engineering 115 (3) (1989) 316–325.

[15] M. Heilig, N. Mallig, T. Hilgert, M. Kagerbauer, P. Vortisch, Large-Scale Application of a Combined Destination and Mode Choice Model Estimated with Mixed Stated and Revealed Preference Data, Transportation Research Record: Journal of the Transportation Research Board (2669) (2017) 31–40.

[16] M. Heilig, N. Mallig, O. Schröder, M. Kagerbauer, P. Vortisch, Implementation of free-floating and station-based carsharing in an agent-based travel demand model, Travel Behaviour and Society ISSN 2214-367X, http://dx.doi.org/10.1016/j.tbs.2017.02.002.

[17] N. Mallig, P. Vortisch, Modeling Car Passenger Trips in mobiTopp, Procedia Computer Science 52 (2015) 938–943, ISSN 18770509.

[18] L. Briem, H. S. Buck, N. Mallig, P. Vortisch, B. Strasser, D. Wagner, T. Zündorf, Integrating public transport into mobiTopp, Procedia Computer Science 109 (2017) 855–860.

[19] H. Bast, D. Delling, A. V. Goldberg, M. Müller–Hannemann, T. Pajor, P. Sanders, D. Wagner, R. F. Werneck, Route Planning in Transportation Networks, in: L. Kliemann, P. Sanders (Eds.), Algorithm Engineering - Selected Results and Surveys, vol. 9220 of *Lecture Notes in Computer Science*, Springer, 19–80, 2016.

[20] B. Strasser, D. Wagner, Connection Scan Accelerated, in: C. C. McGeoch, U. Meyer (Eds.), Proceedings of the 16th Meeting on Algorithm Engineering and Experiments (ALENEX'14), SIAM, 125–137, 2014.

[21] Verband Region Stuttgart, Mobilität und Verkehr in der Region Stuttgart 2009/2010 – Regionale Haushaltsbefragung zum Verkehrsverhalten, 2011.

[22] J. Schlaich, U. Heidl, R. Pohlner, Verkehrsmodellierung für die Region Stuttgart – Schlussbericht, 2011.

Vitae

Lars Briem earned his Bachelor of Engineering in Computer Science at the DHBW in Karlsruhe combined with a job at the SICK AG in 2011. After his studies, he continued to work for SICK AG until he went back to university. During his master studies, he started to give lectures at the DHBW in Karlsruhe in 2015 in software engineering. He finished his Master of Science in Computer Science in 2016 at the KIT in Karlsruhe. Since then, he works as a research assistant at the institute for transport studies at the KIT.

H. Sebastian Buck works as a research assistant at the institute for transport studies at the Karlsruhe Institute of Technology. He studied civil engineering at KIT and earned his Diploma in 2013. His key field of work is traffic engineering. Since 2016 he works on delay analysis and simulation of urban public transport networks.

Nicolai Mallig works as a research assistant at the Institute for Transport Studies at the Karlsruhe Institute of Technology (KIT). He received his degree in Computer Science (Dipl-Inf.) from the University of Freiburg, Germany, in 2003. After working for the German Federal Statistical Office and the Fraunhofer Institute for Systems and Innovation Research ISI he joined the travel demand modelling group at the Institute for Transport Studies in 2011.

Peter Vortisch is a full professor for transportation and the head of the institute for transport studies at the Karlsruhe Institute of Technology. He studied computer science at the University of Karlsruhe and received his PhD in civil engineering from the same university. Before he came back to KIT, he worked for 13 years with PTV Group, a global software and consulting company based in Karlsruhe, Germany, responsible as a vice president for the traffic engineering and traffic management software tools. Dr. Vortisch is a member of the scientific advisory board of the federal minister of transport and digital infrastructure.

Ben Strasser earned his Diploma in computer science at KIT in 2012. Since then, he works as research assistant in the group of Prof. Dr. Wagner at KIT. He is working towards a PhD which is nearly completed. His research interests involve algorithm design and implementation with a focus on route planning and graph partitioning.

Dorothea Wagner heads the Institute of Theoretical Informatics at the Karlsruhe Institute of Technology (KIT). She earned her PhD 1986 from RWTH Aachen, and her habilitation 1992 from TU Berlin. Her research interests are in the field of graph algorithms and computational geometry. Special emphasis lies on the methodology of algorithm engineering with a focus on traffic optimization, network analysis and visualization, network clustering, and energy networks.

Tobias Zündorf earned his Bachelor (2012) as well as his Master (2014) in computer science at KIT. Since then, he works as a research assistant at the Institute of Theoretical Informatics lead by Prof. Dr. Wagner at KIT. His main area of research is algorithm engineering with a focus on route planning and traffic assignments, both for public transit networks.

Highlights

- An integration of the demand side of public transport into a travel demand model mobiTopp
- The integration describes the timetable model, the route search algorithm and the behaviour of agents
- Speed up techniques are used for destination and mode choice