PARALLELIZATION OF TRAVEL DEMAND MODELS WITH DEPENDENT AGENTS: COMPARING MODEL SIMPLIFICATION AND AGENT SYNCHRONIZATION FOR SHARED HOUSEHOLD VEHICLES

Jelle Kübler
Tel: +49 721 608-42256, Email: jelle.kuebler@kit.edu
`https://orcid.org/0000-0001-8095-1815`

Lucas Schuhmacher
Tel: +49 721 608-43465, Email: lucas.schuhmacher@kit.edu
`https://orcid.org/0009-0005-0142-1231`

Robin Andre
Tel: +49 721 608-47772, Email: robin.andre@kit.edu
`https://orcid.org/0009-0003-8717-6411`

Gabriel Wilkes
Tel: +49 721 608-42253, Email: gabriel.wilkes@kit.edu
`https://orcid.org/0000-0002-1027-2734`

Martin Kagerbauer
Tel: +49 721 608-47734, Email: martin.kagerbauer@kit.edu
`https://orcid.org/0000-0003-4252-7874`

Peter Vortisch
Tel: +49 721 608-42255, Email: peter.vortisch@kit.edu
`https://orcid.org/0000-0003-1647-2435`

Address
Karlsruhe Institute of Technology (KIT), Institute for Transport Studies
Kaiserstrasse 12, D-76131 Karlsruhe, Germany

Word Count: 6893 words + 2 table(s) $\times$ 250 = 7393 words

Submission Date: August 28, 2023

ABSTRACT

The increasing complexity and scale of modern agent-based travel demand models pose significant computational challenges in terms of required memory and especially computation time. Improving the efficiency and scalability of these simulations is crucial to produce statistically reliable forecasts for large-scale studies of transportation systems and the effects of policies and measures. Modern approaches from parallel computing have been applied in other agent-based models to evaluate agent decisions in parallel and thereby reduce computation time.

In this paper, we present two approaches to enhance the efficiency of agent-based travel demand models through parallelization. We use the dependency of shared household vehicles and therefore dependent mode choice as a case study. The first approach involves agent synchronization strategies, such as agent grouping by household, and mutual exclusive locks for car allocation in households. The second approach adopts model simplification to reduce inter-agent dependencies, enabling previously dependent agents to be computed in parallel.

Performance evaluations were conducted on a simulated population of 1,871,371 individual persons in the city of Hamburg, Germany. The results demonstrate significant computation time reductions up to a speedup factor of 24.5 when using 32 cores. The model simplification approach, removing all inter-agent dependencies, achieved slightly higher speedups than agent synchronization strategies. Applying the simplified model is shown to produce similar simulation results with no significant differences. Additionally, we explore the transferability of these approaches to scenarios such as ride sourcing or car sharing, showing promising results for applications in this context.

Keywords: agent-based model, travel demand, model simplification, parallel computing, synchronization

INTRODUCTION

Travel demand models are well-established tools for forecasting the effects of transportation policies and measures on travel demand. A recent focus of research has been on agent-based travel demand models (ABTDM) which model travel behavior on the level of individual persons (agents) and thus allow for a higher level of detail than traditional aggregated models (*1*). Since ABTDMs typically apply stochastic models to replicate travel behavior, multiple simulation runs are often necessary to produce statistically significant results. Especially considering the emergence of new mobility services such as bike sharing, car sharing, and ride sourcing, with relatively small user groups, their limited sample size requires repeated simulations. This is also true when investigating inclusive transport solutions for marginalized social groups, like individuals with restricted mobility.

Kwak et al. (*2*) emphasize the importance of computing multiple runs and advise against simulating only a sample of the population, as this significantly increases simulation error. However, the requirement for multiple simulation runs poses a challenge in terms of computation time or available memory.

The trivial parallelization approach of computing multiple independent simulation runs simultaneously requires a high amount of memory, growing linear in the number of simultaneous runs. Therefore, it is desirable to reduce the computation time of a single simulation run through parallelization, which requires no additional memory. Consider a simulation model that takes 25 hours to simulate a single run using a single processing core. Hence, ten simulation runs could be computed in 25 hours using ten cores. However, this would require ten times the amount of memory, since the runs must be independent. By reducing the computation time of a single run to one hour (through parallelization on 32 cores), ten runs can be computed sequentially in ten hours using only the amount of memory of a single run.

In recent years, parallel computing has been applied in various fields of research. The evolution of processors has seen a stagnation in clock speed (determining the speed of computation), hence the focus has shifted to solving problems by using multiple cores or even multiple separate processors in parallel. For this approach, problems are divided into sub-tasks which are computed simultaneously across multiple parallel cores. In this way, parallel computing holds the potential to significantly reduce computation times, paving the way for conducting excessive simulation runs and applying more detailed models to achieve more accurate forecasts.

The practical application of ABTDMs is currently mostly limited to research settings (*3*). Reducing computation times through parallelization can make these models more usable in practice. However, the ever-increasing level of detail in ABTDMs introduces more dependencies and interactions between agents, like sharing vehicles in a household or booking ride sourcing or car sharing services. This complicates the definition of sub-tasks that can be computed independently in parallel. In order to achieve parallel computation of independent tasks, model simplification can be applied to reduce or remove inter-agent dependencies.

In the practice of parallel computing, various techniques such as message passing or shared memory with mutual exclusion have been developed and are now well established (*4*). These techniques can be applied to coordinate parallel task execution in the presence of agent dependencies.

In this paper, we aim to speed up an agent-based travel demand model through parallelization. In doing so, we compare two approaches: model simplification and agent synchronization. Here we focus on inter-agent dependencies when it comes to mode choice i.e. shared household vehicles.

Car availability can be modeled on both the household level (*5–7*) and individual level (*8–12*). Both approaches produce good results, hence we explore the effects of simplifying our model by breaking down car availability on the household level down to the individual level. Using this simplified model, agent decisions can be computed in parallel.

We further explore synchronization through agent grouping and mutual exclusion as used in other fields of research and propose different synchronization strategies for shared household vehicles. Moreover, we investigate how the developed agent synchronization techniques can be applied to scenarios with more inter-agent dependencies (e.g., when modeling ride sourcing or car sharing), where agent-grouping might become infeasible.

LITERATURE REVIEW

To reduce high computation times in agent-based models, several strategies are discussed in the literature. According to Kwak et al. (*2*), a common approach is to simulate only a fraction of the synthetically generated population. However, considering error rates, even fully simulated populations lead to high error rates calling for multiple simulation runs. Therefore, they do not recommend a sample size smaller than 100%. Another approach to speed up the simulation is to reduce the number of activity locations, resulting in a smaller size of choice sets for destination choice. Saleem et al. (*13*) find that computation time grows quadratically with sampled locations, while the bias of simulation results for mode share and travel time is below 2%.

However, both concepts focus exclusively on model simplification and neglect approaches from computer science such as model parallelization, as done by Zhou et al. (*14*). They evaluate the potential of parallel computation of the main activity purpose of household members by using a graphic processing unit (GPU). To account for intra-household interaction, agents from the same household are prioritized to allow for parallel computation of all agents and alternatives. This application of GPU-based parallelization provides a large speedup but is highly dependent on the set of households as well as the alternatives included. However, the agent prioritization technique seems hardly applicable to mode choice especially when including new mobility services such as ride sourcing. Shook et al. (*15*) and Gong et al. (*16*) investigate parallelization of agent-based models for geographical systems. Both apply spatial domain decomposition and synchronization of agents along partitioning borders. Tang et al. (*17*) simulate land use opinion diffusion for spatially related agents. Besides domain decomposition and message passing, they also apply time synchronization between iterations, so agents move through time synchronously.

Dobler (*18*) develops a parallel queue simulation for traffic assignment in MATSim by simulating vehicle queues on nodes or links in parallel. They also synchronize between time steps and between node and link updates to move vehicles to their new queues. Furthermore, a concept for partitioning the road network is proposed which can be used to compute (nearly) independent network sectors in parallel. However, the difficulty of such an implementation is highlighted as it would require dynamically adjusting the partitioning to achieve a balanced workload on all cores. Table 1 shows the related work on parallel agent-based models and their investigated methods.

In the domain of travel demand modeling, agent-based models have been the focus of research in recent years. They can model the travel behavior of individual agents and the effect of policies and measures on that behavior and thereby the resulting travel demand. By modeling a population as individual agents, these models offer the opportunity to model individual agent interactions and dependencies.

One such dependency between agents that complicates model parallelization is the shared

TABLE 1 Overview of Related Work on Parallel ABM and their Investigated Methods.

Related Work	Model Type	Agent Grouping	Agent Sync.	Time Snyc.	Dependent Tasks
Zhou et al. (*14*)	daily main activity (using GPU)	by household	-	no time dimension	×
Shook et al. (*15*)	spatially explicit ABM with CyberGis	spatial domain decomposition	message passing	✓	✓
Gong et al. (*16*)	spatial opinion diffusion	spatial domain decomposition	mutual exclusion	✓	✓
Tang et al. (*17*)	spatial opinion diffusion	spatial domain decomposition	message passing	✓	✓
Dobler (*18*)	traffic assignment MATSim	by link or node	-	✓	×

usage of cars in a household. Since car ownership and car availability are modeled differently across models and some work neglects to adequately differentiate the two concepts, an overview of different approaches is provided. We refer to a person's or household's car ownership as possession of a vehicle derived from surveys and mostly modeled as a discrete choice. However, car ownership alone does not indicate whether the car can be used by an agent, which we refer to as car availability.

A review of different car ownership models is provided, for instance, by Barthelmes et al. (*19*), as well as de Jong et al. (*20*) or Anowar et al. (*21*). In general, with increasing computational power and data availability, models have evolved from aggregated to disaggregated and microscopic (*11, 19*), resulting in state-of-the-art ownership models derived from discrete choice models at either the individual or household level (*11, 22, 23*).

Modeling car availability on the other hand allows for a more realistic representation, as interactions within a household can be considered as well as different degrees of vehicle sharing (*7*). In the ABTDM *mobiTopp*, car availability is modeled at the household level and thus considers the interaction between the agents of a household (*5*). A similar approach is also used by Vovsha et al. (*6*). However, not all car availability models do so. In MATSim, households are optional, leading to car availability being limited to individual agents (*8, 9*). Danalet and Mathys (*10*) model car availability on an individual level too and differentiate between a car being always, on demand, or never available. Loder and Axhausen (*11*) as well as Hillel et al. (*7*) apply a binary distinction of always or never available, although the total number of cars available in the household is nevertheless modeled correctly (*7*). According to Kowald et al. (*12*), an agent's car availability can also be defined as the ratio of cars to agents with a driver's license within a household being greater or equal to 0.5.

To the best of the authors' knowledge, parallel computing has not yet been applied to ABTDMs with dependencies in mode choice. The literature review shows that parallel computing has been applied to the field of ABTDM for modeling household main activities (*14*) and traffic assignment (*18*). In both cases, agents can be grouped in a way that the resulting parallel tasks are independent. Therefore, in this paper, we explore strategies for agent synchronization, when it is not possible to define independent tasks.

APPROACH

For this paper, we apply a simplified version of the agent-based travel demand modeling framework *mobiTopp* (*5, 24, 25*) focusing on a few core aspects relevant to evaluating the effects of parallelization. The *mobiTopp* framework consists of two modules: a long-term module which generates a synthetic population and models long-term decisions, and a short-term module which simulates the travel demand of that population for a whole week. The population is synthesized from a representative sample provided by a national household travel survey (*26*) and regional socio-demographic statistics using an iterative proportional updating algorithm (*27*). Additionally, the long-term module assigns mobility tools including personal bicycles, driver's licenses, transit passes, and shared household cars. Finally, each agent is assigned a fixed workplace or educational institution and a week activity schedule (*28*). In the short-term module, the activities and the intermediate trips of all agents are simulated including the choice of destination and mode of travel.

Destination and mode choice are determined by applying discrete choice models: for this paper multinomial logit models are employed as described in the *Model Simplification* Section. The simulation is based on traffic assignment zones, which act as home zones for households and as destinations for trips. Travel demand is determined as origin-destination trip counts per mode, hence no route assignment is computed. For this paper, the choice set of supported modes is *pedestrian*, *bike*, *public transport*, *car* and *car as passenger*.

In every choice situation, this full choice set is filtered in the following way so it only contains the available modes: Agents without a driver's license or personal bike can never use *car* or *bike* respectively. Further, when departing from home, *car* is only available if at least one household car is currently at home. Finally, when not at home, persons departing with a fixed mode (*car* and *bike*) must use it until they return home. On the other hand, agents using flexible modes (*pedestrian*, *public transport* and *car as passenger*) may only use flexible modes until they return home. After determining their availability, the utility of each remaining mode is computed and the choice is determined by evaluating a multinomial logit model. Finally, if the mode *car* was chosen, the agent takes a household car. This general procedure is shown on the top in Figure 1.

In the following Figures 1 to 6, the mode choices of the shown agents are assumed to evaluate to *car*; also their (initial) car availability check is assumed to be positive. In these figures, T denotes the set of simulation steps which are iterated in chronological order and P denotes the set of all person agents. $t_i \in T$ is a single time step containing a set of agents (either persons p or households h). An exemplary person i of household j is denoted by $p_i^{h_j}$.

In this paper, we focus on parallelization strategies for the short-term module. Efficient algorithms for the long-term module will be part of future research and therefore omitted in this paper. We focus on the dependency of shared household cars and their effect on mode choice. The choice of activity is a long-term decision and not part of the short-term simulation we aim to optimize through parallelization. Also, destination choice is assumed to be independent in this paper. We investigate two approaches to achieve a parallel demand simulation: first, we develop multiple strategies for agent synchronization to handle dependencies between agents during parallel computing. Second, we employ model simplification to remove these dependencies by modeling car availability on a personal level instead of a household level.

Agent Synchronization

In general, the strategy behind parallel computing is dividing a large problem into smaller sub-problems (or tasks). These are then dispatched onto multiple cores (either in the form of processes or threads) and computed in parallel. Once all sub-problems have been computed, the results are combined to form the solution of the original, larger problem (*29*).

In the context of this paper, the paramount objective is to determine the travel demand of a population within a survey area. This can be broken down into tasks on the agent level: i.e. determining the travel demand of a single person over one week. This can be further broken down into computing individual choices made by agents including the choice of destination and travel mode. In this section, we present synchronization strategies for mode choice and explore the transferability of these methods in the *Evaluation* Section.

Despite breaking down the travel demand simulation into the task of computing individual choices, these tasks are still interdependent since members of the same household share household vehicles and could allocate the same car when computed in parallel. This critical code section of the choice process is highlighted in Figure 1.

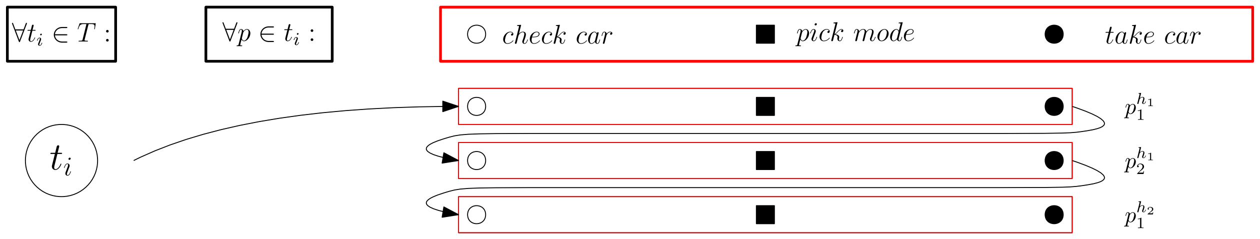

FIGURE 1 A sequential execution of person mode choices. The critical code section (marked red) may cause race conditions if executed in parallel without proper synchronization.

Moreover, two choices of two agents $p_1^{h_1}$ and $p_2^{h_1}$ in the same household can be interdependent even if they take place at different times. If agent $p_1^{h_1}$ departs from home on Monday morning, taking the car, then agent $p_2^{h_1}$ departing on Monday afternoon won't have access to that car. Simulating the mode choice of agent $p_2^{h_1}$ before the choice of agent $p_1^{h_1}$ would result in an inconsistent state. Therefore, all agents must be simulated synchronously through time. This means that all choices happening at the same point in time must be computed before all later events. In our simulation, a discrete time step of one minute is used. Within each time step, all choices may be computed in parallel. However, the next time step can only be started once all choices of the current time step have been made. A synchronization barrier is used, waiting for all cores to complete their tasks, before the tasks of the next time step are dispatched. In the following, we will look at the parallelization of tasks (i.e. mode choices) within a single time step.

A first intuitive approach, called agent-grouping (*30*), is to identify interdependent agents and group them such that no two groups contain mutually dependent agents. For the shared household car dependency, agent-grouping is trivial, since each agent is part of exactly one household. Thus, it is possible to dispatch households to cores. Within each household, the choices of agents are computed sequentially, while independent households are processed in parallel as shown in Figure 2. In this way, no car can be allocated by two agents simultaneously and therefore no synchronization is required.

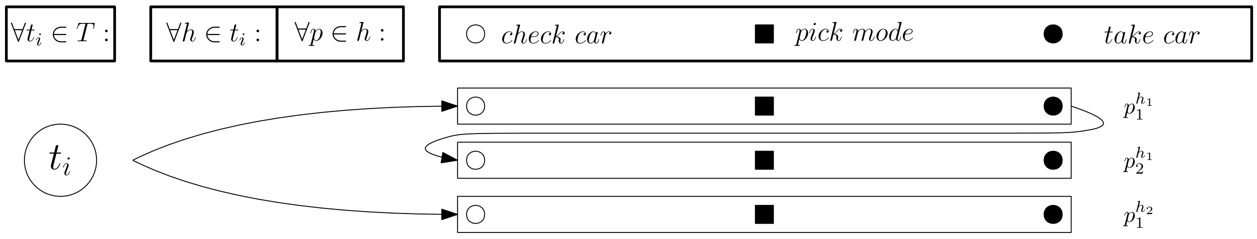

FIGURE 2 Household grouping: households are processed in parallel while its members are processed sequentially on the same core.

Grouping by household produces relatively small groups: according to UnitedNations (*31*) households with 6 or more members only account for a small share in most countries. E.g., in 2010 over 90% of the households in the U.S. had five or fewer members (*31*).

However, this technique is hardly transferable to shared vehicles or agent dependencies in general. For instance, when examining car sharing, all agents departing from the same zone are dependent in that they all might request the same car sharing vehicle. Grouping agents by departure zone can create arbitrarily large groups especially if there are multiple car sharing providers and agents with more than one membership. This however diminishes the potential of parallelization, since large sets of dependent agents need to be processed sequentially.

When dependency structures between agents grow too complex to compute independent groups, or when dependent groups grow too large, causing the parallelization effect to vanish, applying agent grouping becomes unsuitable. As an alternative, we can explore parallelization strategies with interdependent tasks that require agent synchronization. In the following, we will explore synchronization strategies for interdependent tasks. We will describe and explain them exemplary for the dependency of shared household vehicles, however, these techniques are designed to be applicable to other inter-agent dependencies such as ride sourcing and car sharing services (which we will investigate in the *Evaluation* Section). When working with interdependent tasks on multiple cores, two major strategies are prominent in practice: message passing and shared memory. For this paper, we apply synchronization through shared memory as it does not require an additional program communication structure to maintain simulation consistency.

In the context of this paper, the shared memory that is accessed by interdependent parallel tasks is a list of available shared household vehicles. Once an agent (departing from home) starts to decide which mode to use, the availability of a household car is checked. If available, the utility for using the car is computed alongside the utility of all other available modes. Using a logit model, one of the available modes is selected. If the selected mode is *car*, one of the household cars is allocated by the agent.

If not handled carefully, computing this vehicle allocation process in parallel can lead to so-called race conditions. Assume the following situation: two agents of the same household are processed simultaneously on two parallel cores. Both check the availability of household vehicles and both perceive one available car. After both have checked the availability, they evaluate the utilities and logit function, and both decisions result in using the mode *car*. Since both agents now try to allocate a car, one of them will end up with an empty set of available cars resulting in an inconsistent state, and the program is terminated. This critical code section is marked yellow in Figure 3.

To avoid these kinds of race conditions, so-called "mutual exclusive locks" (mutex lock) can be applied to guard the critical shared memory from parallel access. Mutex locks are a common technique for synchronization in parallel programming practice. They are used to control access to shared resources or critical code sections. Before accessing such a shared resource, respectively before entering a critical code section, cores must acquire the lock, which can only be held by a single task at a time. When a task tries to acquire a free mutex lock, it becomes the exclusive owner of the lock. Other tasks can no longer acquire the lock until it is released. If a mutex lock is already owned by another task, the requesting task has to wait until the lock is released.

As explained in the race-condition example above, it must not be allowed to change the set of available cars between checking their availability and the final mode choice decision and car allocation of an agent. This is a critical code section and may not be executed by more than one (dependent) agent simultaneously.

As a first attempt, we can apply a mutex lock to this critical code section. Since not all agents are dependent, we can apply individual locks for each household. In this way, only one agent per household may enter this synchronized, critical code section and perform its mode choice as shown in Figure 3. This extensive synchronization, however, now replicates the behavior of the agent-grouping strategy: effectively all mutually dependent agents are computed sequentially, as they mutually block each other through their mutex lock.

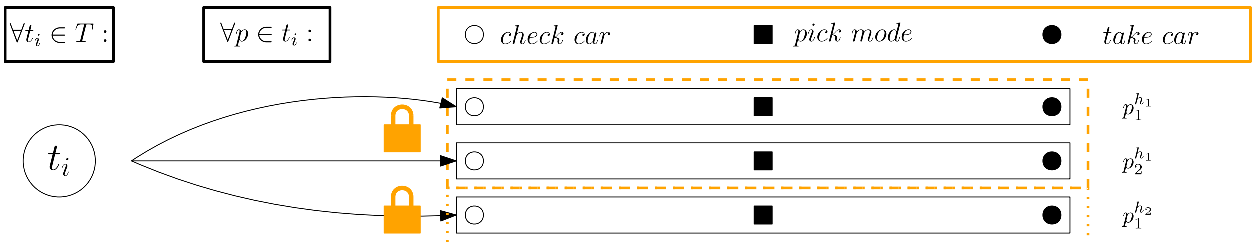

FIGURE 3 Parallel persons: all agents are processed in parallel, only one member per household may enter the locked critical code section.

To achieve higher parallelization, the expensive computation of mode utilities and evaluation of the mode choice logit function should be moved outside the synchronized code section so it can be computed in parallel. Only the actual car allocation is critical as it changes the shared memory. However, the mode choice model requires to evaluate the availability of shared household cars. By moving the choice computation and the availability check out of the synchronized code section, we can no longer assume the previously determined car availability to be correct. Hence, the availability has to be checked again inside the synchronized code section before the car is actually allocated. If the car is no longer available because it was taken in the meantime, mode choice has to be evaluated again without the alternative *car*. This way, the first availability check without locking can be used as an availability estimation. Assuming that such conflicts only happen rarely, the cost of recomputing some mode choices should be outweighed by the benefit of computing all other choices in parallel. This approach is visualized in Figure 4.

Model Simplification

As a second approach, we employ model simplification to reduce inter-agent dependencies. By removing these dependencies, agent-agent interactions do not have to be resolved subsequently

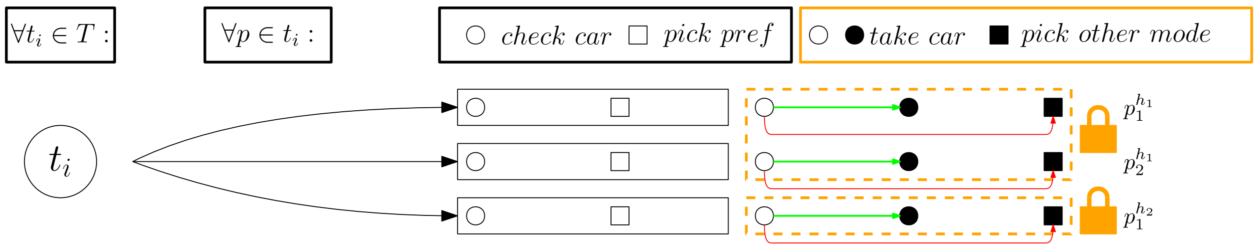

FIGURE 4 Parallel preference: all agents are processed in parallel, only one member per household may enter locked critical code section. A preferred mode is computed in parallel, merely the actual car allocation is synchronized. If no car is available during allocation, mode choice is reevaluated.

but can be avoided entirely. This allows for independent agents and consequently separate tasks to be computed independently on different cores in parallel.

Currently, car availability in *mobiTopp* is modeled at the household level (*5*). The underlying car ownership is modeled in the long-term module: the number of owned cars in a household is determined using a logit model which is calibrated with a representative sample provided by a large national household survey (*26*). During the short-term simulation, when agents depart from home using a car, it is made unavailable for other household members until the agent returns home. In this way, car availability is modeled and updated dynamically with each departing and arriving household member. As described in the section above, this is the cause for dependencies between tasks on parallel cores.

As part of the model simplification, we abandon the accurate tracking of available household cars. The number of owned household cars is still computed, since it affects other models of the long-term module, such as the transit pass ownership or activity schedule generation. However, in the simulation, agents no longer perform car allocation. This means agents can still select the mode *car* in mode choice but the household cars are never moved, hence they do not need to be synchronized as shown in Figure 5.Instead of modeling detailed car allocation, we introduce a person-specific, discrete car availability attribute. Similar to Danalet and Mathys (*10*), we distinguish whether a car is never, partially, or always available for an agent. This is determined as follows:

- If there are no cars in a household, a car cannot be used by any agent of that household and is therefore never available.
- If there are at least as many cars in a household as agents with a driver's license, the car is always available for all these agents.
- If there are more agents with a driver's license than cars in a household, the car is defined as only partially available to all agents in that household. Therefore, we are more restrictive than Kowald et al. (*12*), as described in the *Literature* Section.

This adjustment alone would lead to a significant overestimation of car use as driver and passenger in the mode choice model. Therefore, we adjusted and recalibrated the multinomial logit mode choice model to reproduce the original modal split. Thus, new parameters for the modes *car* and *car as passenger* were integrated to account for partial availability. Since the impact of the adjusted car availability on the other modes is marginal in our model, no further changes had to be made. This approach enables the adaption of existing models for parallel computation without having to develop an entirely new mode choice model. Also, this allows for better comparability of the results with the base model, since only minor changes were made.

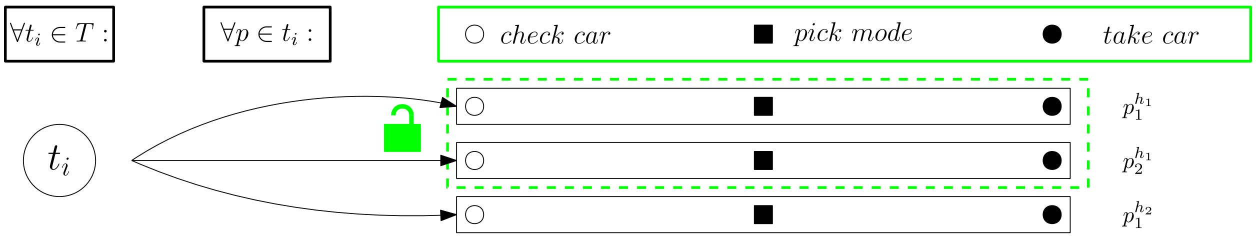

FIGURE 5 Parallel persons: no dependencies and no mutex locks due to model simplification.

The main benefit of the model simplification is to enable a different parallelization strategy. With inter-agent dependencies, the simulation has to synchronize at predetermined time steps to ensure a consistent state - the agents' actions are processed in correspondence to their occurrence time ($S_{TimeSync}$). With the removal of dependencies, each agent can be simulated completely independent of the actions of other agents. The full independent (S_{Indep}) strategy processes over the agents rather than the time slices. Figure 6 highlights the operational difference to the other time-synchronized algorithms.

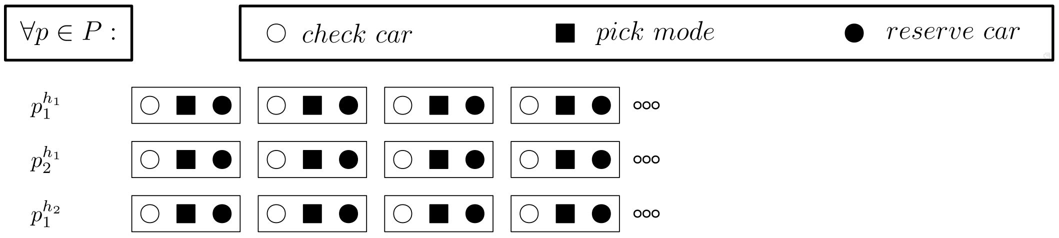

FIGURE 6 Fully independent execution: The entire activity plan of each agent is simulated in parallel without any synchronization at time steps.

EVALUATION AND RESULTS

For our evaluation, we apply the developed models to the city of Hamburg in northern Germany. The survey area consists of 2715 traffic assignment zones and the simulated city population consists of 1,041,545 households and 1,871,371 individual persons. The simulation period is set to a single day, as it suffices to observe the effects of the parallelization. In the following, we present the methodology applied to evaluate the effect of the developed parallelization strategies on both the computation time as well as the quality of the simulation results.

Computation time is evaluated for both the agent synchronization strategies *household grouping* (O_{Hh}), *synchronized persons* (O_{Person}), *synchronized retry* (O_{Retry}) as well as the model simplification strategies *time synchronized* ($S_{TimeSync}$) and *full independent* (S_{Indep}) as described above. To measure their scalability, all strategies were simulated with 4, 8, 16, and 32 cores. In the following, a combination of strategy and core count is called a configuration. The simulated configurations and the measured times are listed in the right half of Table 2. All configurations were executed on a high-performance workstation with the following specification: AMD EPYC 7374 processor (16 physical, 32 logical cores) with 3.20 GHz base speed (4.7 GHz effective speed), and 256 GB RAM at 3,2 GHz.

TABLE 2 Overview of Evaluated Strategies and Measured Times over #Cores.

Name	Choice	Parallel	Synchronize by	1	4	8	16	32
O_{Seq}	original	-	-	25.8 h	-	-	-	-
O_{Hh}	original	household	time	-	6.8 h	3.4 h	2.1 h	1.4 h
O_{Person}	original	person	household + time	-	6.8 h	3.4 h	2.1 h	1.4 h
O_{Retry}	original	preferred mode	household + time	-	6.3 h	3.4 h	2.2 h	1.4 h
S_{Seq}	simple	-	-	25.8 h	-	-	-	-
$S_{TimeSync}$	simple	person	time	-	5.8 h	3.3 h	2.1 h	1.4 h
S_{Indep}	simple	person	-	-	5.5 h	3.1 h	1.6 h	1.1 h

For comparison a *sequential baseline* simulation (with and without model simplification) was computed using a single core (O_{Seq}, S_{Seq}). The observed variance of computation time is insignificant. Therefore, only the mean computation time of each configuration will be considered in the following.

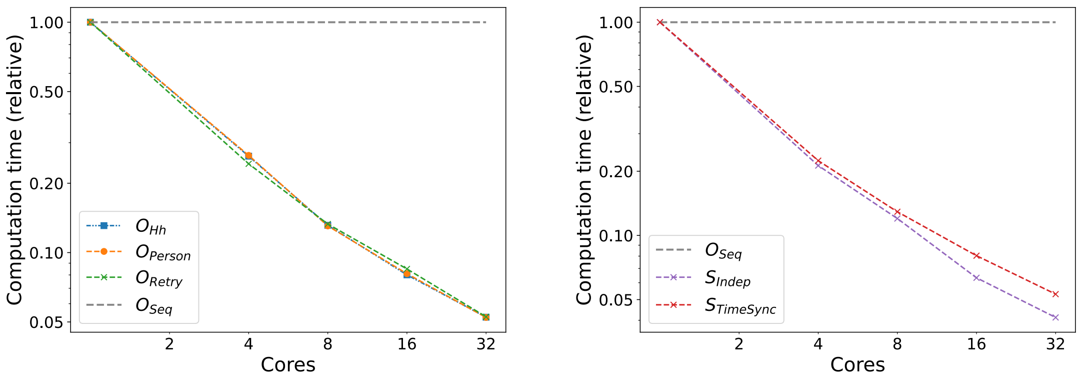

FIGURE 7 Computation time reduction relative to sequential execution for agent synchronization (left) and model simplification (right).

Figure 7 shows the results of the performance measurements for both the agent synchronization (on the left) and model simplification (on the right) configurations compared to the sequential execution. In both, the execution time is normalized by the sequential execution time. They show that applying any of the parallelization techniques significantly reduces the simulation time down to 4-5% of the original computation time when using 32 cores. The three presented agent synchronization strategies O_{Hh}, O_{Person} and O_{Retry} show only little difference. S_{Indep} shows a slightly higher reduction in computation time compared to $S_{TimeSync}$.

Figure 8 compares the performance of the *household grouping* strategy using the original model (O_{Hh}) and the *full independent* execution of the simplified model (S_{Indep}). On the left,

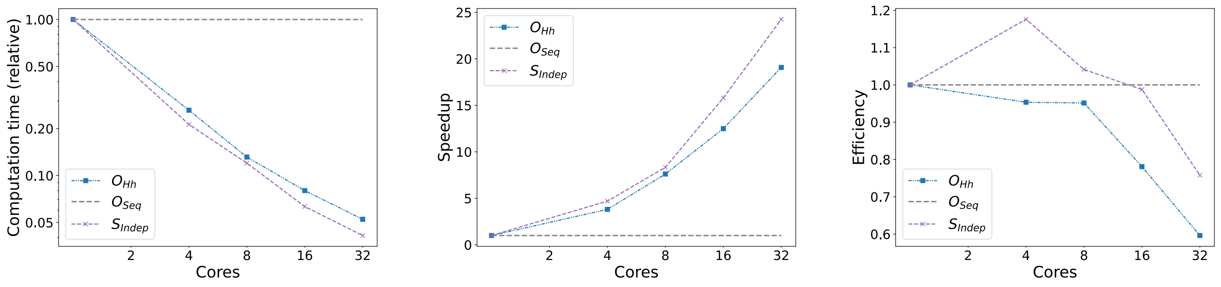

FIGURE 8 Comparison of agent synchonization (O_{Hh}) and model simplification (S_{Indep}) by computation time (left), speedup (mid) and efficiency (right).

Figure 8 shows that the *full independent* execution produces a slightly higher reduction in computation time. It is between 19% and 24% faster (except for 8 cores where it is only 9% faster). In the middle, Figure 8 shows the resulting speedup (parallel computation time divided by sequential computation time). Up to 8 cores S_{Indep} has a slightly higher speedup than O_{Hh} but visibly outperforms the latter for 16 and 32 cores. For 32 cores, O_{Hh} achieves a speedup of 18.4 while S_{Indep} reaches a speedup of 24.5. On the right, Figure 8 shows the efficiency (speedup divided by the number of cores) of O_{Hh} and S_{Indep}. *Household grouping* O_{Hh} has a very high efficiency near 1.0 up to 8 cores, which then drops down to 0.58 for 32 cores. The *full independent* execution S_{Indep} shows a remarkably high efficiency even surpassing the *sequential baseline* of 1.0. This is possible because the *full independent* execution abandons the time synchronization of agents which was used in the sequential execution. For 32 cores, the efficiency of S_{Indep} drops to 0.73.

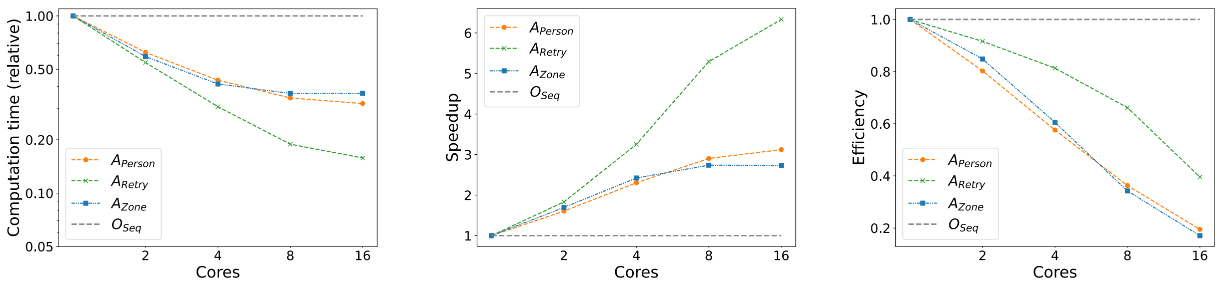

FIGURE 9 Agent synchronization strategies applied to artificially amplified dependencies: computation time (left), speedup (mid) and efficiency (right).

To estimate the transferability of our approach and to differentiate the developed agent synchronization strategies, we introduced additional, artificial dependencies between agents in the same home zone (to emulate, e.g., some kind of artificial car sharing or ride sourcing). For this artificial highly dependent population we again simulate a sequential baseline simulation (A_{Seq}) and the three strategies: *zone grouping* (A_{Zone}), *synchronized persons* (A_{Person}), *synchronized retry* (A_{Retry}). The relative computation time, speedup, and efficiency of the amplified dependency experiment are shown in Figure 9 for different core counts. It shows that the *synchronized retry* (A_{Retry}) strategy significantly outperforms the other two approaches with a speedup factor of 6.33 compared to 3.12 and 2.73. The amplification of dependencies noticeably reduces the maximum speedup that can be achieved compared to only intra-household dependencies.

In addition to the impact on performance, we evaluate the bias of the simulation results caused by the model simplification. To do so, we compare the simulation results of the original model (O_{Seq}) with the results of the simplified model (S_{Seq}) with regard to four key measurements: modal split, mileage, travel distance distribution, and travel time distribution as shown in Figure 10. All four plots show only little differences between the original and the simplified model. Applying a statistical t-test with a significance threshold of $p = 0.05$ suggests that there is no significant difference between the simulation results of the original and simplified model regarding all four key measurements. Analogously, the parallel executions of the original and simplified model also show no significant differences in the simulation results compared to the sequential executions.

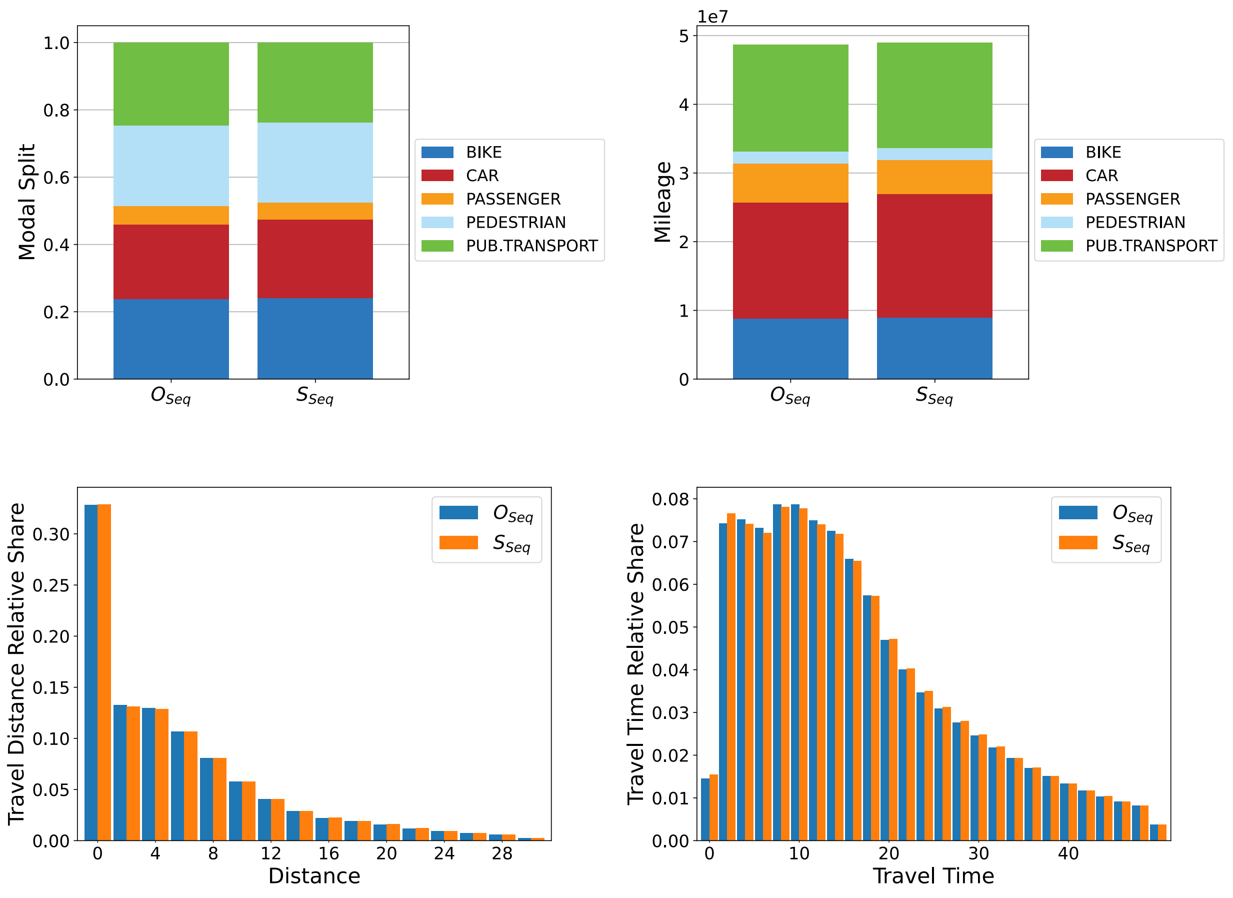

FIGURE 10 Comparison of original and simplified model results: modal split (top left), mileage (top right), travel distance (bottom left) and travel time (bottom right).

CONCLUSION

The results demonstrate that applying parallelization to an ABTDM significantly reduces computation time. With the most efficient strategy of *full independent* execution, the computation time of a single simulation run could be reduced from 25.8 to 1.1 hours. Hence, 24 parallelized simulation runs can be computed within the original simulation time.

The *retry* strategy (O_{Retry}) presented in this paper cannot be trivially ported to nested logit models. The mode availability requires additional care to ensure unbiased mode choice since alternatives in nest groups are not independent. An estimated car availability can impact the selection probability of related modes. In case the estimation is no longer valid during mode allocation, mode choice has to be reevaluated even if a non-conflicting mode was chosen. In the future, this synchronization strategy could be expanded to also be compatible with models containing dependent alternatives.

When analyzing the dependency of shared household vehicles, our results show that all strategies exhibit a similar reduction in computation time. However, applying a simplified model that removes all agent dependencies, makes the computation slightly faster than the other strategies. Therefore, we suggest applying a *full independent* execution strategy if and only if all inter-agent dependencies can be removed (including dependencies spanning multiple time steps). The simplified model shows no benefit over the synchronization strategies when time synchronization is not removed. It is worth noting that utilizing parallelization has a detrimental effect on reproducibility since the agent execution order can fluctuate. However, between two simulation runs with identical input data, no significant differences could be observed. However, this could be worth looking into in future research, especially when investigating scenarios with extensive inter-agent dependencies.

Comparing the simulation results of the simplified model to the original model showed no significant differences, as confirmed by a statistical t-test. Thus, when trying to reduce the complexity of ABTDMs, it can be recommended to model car availability on the individual level rather than the household level. Even though the proposed model simplification technique might not be transferable to scenarios involving sharing vehicles, the simplified household car availability model can still be used in more complex ABTDMs. Suppose the aspect of household vehicles is not investigated in detail. In that case, model complexity can be reduced in this aspect, enabling focus on other aspects where greater attention and detail are needed (e.g., car sharing models sensitive to implemented measures).

Finally, we estimate the transferability of the presented synchronization strategies by artificially amplifying the number of inter-agent dependencies. Here, our results show the highest speedup when computing preferred modes in parallel and only synchronizing the actual car allocation (potentially reevaluating mode choice). As expected, allocation conflicts and, therefore, reevaluation of mode choice occur rarely enough so that the benefit of computing mode preferences in parallel outweighs the cost of reevaluation. These promising results indicate that these strategies also apply to, e.g., ride sourcing and car sharing dependencies between agents.

In future research, we want to extend the investigated scenarios and introduce situations with sharing and hailing vehicles. In this way, we aim to check the hypothesis presented above that the proposed synchronization is also applicable and efficient when additional inter-agent dependencies are added, even beyond the scope of households. Finding ways to resolve car sharing dependencies through model simplification without significantly distorting the simulation results is also worth exploring.

Moreover, other types of dependencies (other than shared vehicles), such as joint trips or activities, could be investigated. In those cases, both the process of joint decision-making as well as an efficient parallel implementation are of interest. If these joint decision-making dependencies can also be computed in parallel, a general concept for inter-agent dependency synchronization could be developed.

AUTHOR CONTRIBUTION STATEMENT

The authors confirm contribution to the paper as follows: study conception and design: J. Kübler, L. Schuhmacher, R. Andre, G. Wilkes, M. Kagerbauer, P. Vortisch; data collection: R. Andre, J. Kübler; analysis and interpretation of results: J. Kübler, R. Andre, L. Schuhmacher; model implementation: J. Kübler, R. Andre; draft manuscript preparation: J. Kübler, L. Schuhmacher, R. Andre. All authors reviewed the results and approved the final version of the manuscript.

ACKNOWLEDGEMENTS

This study is part of the project #transmove at the DigiLab of the State Agency for Roads, Bridges, and Waterways (LSBG) for the Free and Hanseatic City of Hamburg. It is funded under the "Digitalization of Municipal Transport Systems" grant program by the Federal Ministry for Digital and Transport (BMDV) (Grant Reference Number: 16DKV41086). The model is provided by courtesy of MOIA.

REFERENCES

1. Shiftan, Y. and M. Ben-Akiva, A practical policy-sensitive, activity-based, travel-demand model. *The Annals of Regional Science*, Vol. 47, 2011, pp. 517–541.
2. Kwak, M.-A., T. Arentze, E. de Romph, and S. Rasouli, Activity-based dynamic traffic modeling: Influence of population sampling fraction size on simulation error. In *13th International Conference on Travel Behaviour Research (IATBR 2012), July 15-20, 2012, Toronto, Canada*, 2012, pp. 1–17.
3. Crooks, A. T. and A. J. Heppenstall, Introduction to agent-based modelling. In *Agent-based models of geographical systems*, Springer, 2011, pp. 85–105.
4. Skillicorn, D. B. and D. Talia, Models and languages for parallel computation. *Acm Computing Surveys (Csur)*, Vol. 30, No. 2, 1998, pp. 123–169.
5. Mallig, N., M. Kagerbauer, and P. Vortisch, mobiTopp – A Modular Agent-based Travel Demand Modelling Framework. *Procedia Computer Science*, Vol. 19, 2013, pp. 854–859, the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).
6. Vovsha, P., J. Freedman, V. Livshits, and W. Sun, Design Features of Activity-Based Models in Practice: Coordinated Travel–Regional Activity Modeling Platform. *Transportation Research Record*, Vol. 2254, No. 1, 2011, pp. 19–27.
7. Hillel, T., J. Pougala, P. Manser, R. Luethi, W. Scherr, and M. Bierlaire, Modelling mobility tool availability at a household and individual level: A case study of Switzerland. In *hEART conference. Lyon, France*, 2020.
8. Ciari, F., M. Balmer, and K. W. Axhausen, Mobility tool ownership and mode choice decision processes in multi-agent transportation simulation. ETH Zurich, IVT, Zurich, 2007, 7th Swiss Transport Research Conference (STRC 2007); Conference Location: Ascona, Switzerland; Conference Date: September 12-14, 2007.
9. Horni, A., K. Nagel, and K. Axhausen (eds.) *Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London, 2016.
10. Danalet, A. and N. Mathys, Mobility Resources in Switzerland in 2015. In *Swiss Transport Research Conference. Ascona, Switzerland*, 2018.

11. Loder, A. and K. W. Axhausen, Mobility tools and use: Accessibility's role in Switzerland. *Journal of Transport and Land Use*, Vol. 11, No. 1, 2018, pp. 367–385.

12. Kowald, M., B. Kieser, N. Mathys, and A. Justen, Determinants of mobility resource ownership in Switzerland: changes between 2000 and 2010. *Transportation*, Vol. 44, No. 5, 2017, pp. 1043–1065.

13. Saleem, M., O. B. Västberg, and A. Karlström, An activity based demand model for large scale simulations. *Procedia computer science*, Vol. 130, 2018, pp. 920–925.

14. Zhou, H., J. Dorsman, M. Snelder, E. de Romph, and M. Mandjes, GPU-based parallel computing for activity-based travel demand models. *Procedia Computer Science*, Vol. 151, 2019, pp. 726–732.

15. Shook, E., S. Wang, and W. Tang, A communication-aware framework for parallel spatially explicit agent-based models. *International Journal of Geographical Information Science*, Vol. 27, No. 11, 2013, pp. 2160–2181.

16. Gong, Z., W. Tang, D. A. Bennett, and J.-C. Thill, Parallel agent-based simulation of individual-level spatial interactions within a multicore computing environment. *International Journal of Geographical Information Science*, Vol. 27, No. 6, 2013, pp. 1152–1170.

17. Tang, W., D. A. Bennett, and S. Wang, A parallel agent-based model of land use opinions. *Journal of Land Use Science*, Vol. 6, No. 2-3, 2011, pp. 121–135.

18. Dobler, C., Implementation of a time step based parallel queue simulation in MATSim. In *10th Swiss Transport Research Conference. Monte Verita, Ascona*, 2010.

19. Barthelmes, L., M. Heilig, C. Klinkhardt, M. Kagerbauer, and P. Vortisch, The effects of spatial characteristics on car ownership and its impacts on agent-based travel demand models. *Procedia Computer Science*, Vol. 201, 2022, pp. 296–304, the 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40).

20. de Jong, G., J. Fox, A. Daly, M. Pieters, and R. Smit, Comparison of car ownership models. *Transport Reviews*, Vol. 24, No. 4, 2004, pp. 379–408.

21. Anowar, S., N. Eluru, and L. F. Miranda-Moreno, Alternative Modeling Approaches Used for Examining Automobile Ownership: A Comprehensive Review. *Transport Reviews*, Vol. 34, No. 4, 2014, pp. 441–473.

22. Paredes, M., E. Hemberg, U.-M. O'Reilly, and C. Zegras, Machine learning or discrete choice models for car ownership demand estimation and prediction? In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2017, pp. 780–785.

23. Bhat, C. R. and S. Sen, Household vehicle type holdings and usage: an application of the multiple discrete-continuous extreme value (MDCEV) model. *Transportation Research Part B: Methodological*, Vol. 40, No. 1, 2006, pp. 35–53.

24. Mallig, N. and P. Vortisch, Modeling travel demand over a period of one week: The mobiTopp model. *arXiv preprint arXiv:1707.05050*, 2017.

25. Kübler, J., L. Briem, and N. Mallig, *mobiTopp: https://github.com/kit-ifv/mobitopp*, 2022.

26. Nobis, C. and T. Kuhnimhof, *Mobilität in Deutschland 2017 - Mobility in Germany 2017*. BMVI, infas, DLR, IVT, infas 360., Bonn, Berlin, 2018.

27. Ye, X., K. Konduri, R. M. Pendyala, B. Sana, and P. Waddell, A methodology to match distributions of both household and person attributes in the generation of synthetic pop-

ulations. In *88th Annual Meeting of the transportation research Board, Washington, DC*, 2009.

28. Hilgert, T., M. Heilig, M. Kagerbauer, and P. Vortisch, Modeling week activity schedules for travel demand models. *Transportation Research Record*, Vol. 2666, No. 1, 2017, pp. 69–77.
29. Quinn, M. J., *Parallel computing theory and practice*. McGraw-Hill, Inc., 1994.
30. Delhoum, Y., R. Belaroussi, F. Dupin, and M. Zargayouna, Activity-based demand modeling for a future urban district. *Sustainability*, Vol. 12, No. 14, 2020, p. 5821.
31. UnitedNations, Household size and composition around the world, 2017. *Data booklet, http://www. un. org/en/development/desa/population/publications*, 2017.