Transportation Research Part C 98 (2019) 370–390

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Data-driven activity scheduler for agent-based mobility models

Jan Drchal*, Michal Čertický, Michal Jakob

Artificial Intelligence Center, Czech Technical University in Prague, Karlovo náměstí 13, Prague 2, 12135, Czech Republic

ARTICLE INFO

Keywords:
Activity-based model
Agent-based model
Machine learning
Travel demand model
Population modelling
Model validation

ABSTRACT

Activity-based modelling is a modern agent-based approach to travel demand modelling, in which the transport demand is derived from the agent's needs to perform certain activities at specific places and times. The agent's mobility is considered in a broader context, which allows the activity-based models to produce more realistic trip chains, compared to traditional trip-based models. The core of any activity-based model is an activity scheduler – a software component producing sequences of agent's daily activities interconnected by trips, called activity schedules. Traditionally, activity schedulers used to rely heavily on hard-coded knowledge of transport behaviour experts. We introduce the concept of a Data-Driven Activity Scheduler (DDAS), which replaces numerous expert-designed components and their intricately engineered interactions with a collection of machine learning models. Its architecture is significantly simpler, making it easier to deploy and maintain. This shift towards data-driven, machine learning based approach is possible due to increased availability of mobility-related data. We demonstrate DDAS concept using our own proof-of-concept implementation, perform a rigorous analysis and compare the validity of the resulting model to one of the rule-based alternatives using the Validation Framework for Activity-Based Models (VALFRAM).

1. Introduction

Activity-Based Models (ABM) (Ben-Akivai et al., 1996) are a specific kind of agent-based models of travel demand (Castiglione et al., 2015). In contrast to traditional trip-centred alternatives, such as four-step demand model (McNally, 2008), activity-based models employ so-called activities and their sequences to represent the transport-related behaviour of the population. Travel demand is then occurring due to the necessity of the agents to satisfy their needs through activities performed at different places at different times. Trip origins, destinations and times are endogenous outcomes of activity scheduling. The activity-based approach considers individual trips in context and therefore allows representing realistic trip chains.

A fundamental unit on the output of ABM is a sequential *activity schedule* representing the behaviour of a single agent during a specified time period (usually one day). An activity schedule consists of a list of *activities* and *trips* connecting them. Both the activities and trips are fixed in time and space – they have a specific start time and duration, as well as the activity location or trip's origin/destination/route. Also, each activity has a specific activity type (e.g., `work`, `school` or `shop`) and each trip has its mode of transport (e.g. `car` or `walk`).

The general workflow of ABMs, as depicted in Fig. 1, can be described by three steps:

1. synthesize a population of the modelled area,

* Corresponding author.
E-mail addresses: drchajan@fel.cvut.cz (J. Drchal), certimic@fel.cvut.cz (M. Čertický), michal.jakob@fel.cvut.cz (M. Jakob).

https://doi.org/10.1016/j.trc.2018.12.002
Received 17 May 2018; Received in revised form 10 December 2018; Accepted 11 December 2018
Available online 24 December 2018
0968-090X/

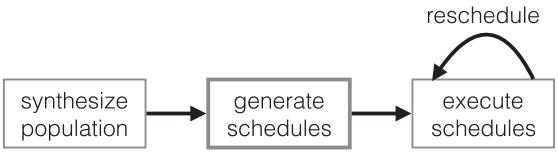


Fig. 1. An overview of Activity-Based Model (ABM) workflow. A population of agents corresponding to the population of modelled system is synthesized, a schedule is generated for each agent and finally, the schedules are executed in a simulation and modified if necessary.

2. plan an activity schedule for each member of the population, and
3. simulate the execution of these activity schedules.

In this paper, we focus on the second step, in which the schedules are generated. A software responsible for the generation of activity schedules is called *activity scheduler*.

More specifically, we focus on *data-driven activity scheduling* in this paper. While most current activity schedulers use machine learning components to some extent, they still heavily depend on expert knowledge, implemented by hard-coded rules. This is a result of the limited availability of data on human behaviour in the past. However, increasing mobile device penetration rate in the present day allows for a cheap large-scale collection of transport-related data (Gadzinski, 2018; Alsger et al., 2018). Global services, such as Google Maps Timeline[1] collect big datasets, which could be utilized by a new generation of data-driven activity schedulers. While privacy policy constraints are the most prominent obstacles in the utilization of such data, anonymization approaches such as those used in Open PFLOW (Kashiyama et al., 2017) can make their adoption possible. We argue that a wider adoption of data-driven approach can significantly simplify the activity scheduling process, since the complexity shifts from many expert-designed components and their intricately engineered interactions to a collection of parameters of machine learning models trained automatically using the available data. This simplification reduces an effort required for model maintenance and deployment. Also, in the presence of a large amount of available training data, there is a good chance that data-driven models will outperform actual human experts, similarly to what happened with image classification (He et al., 2015), speech recognition (Xiong et al., 2016) or games (Silver et al., 2017; Moravčík et al., 2017). The significance of machine learning in topics related to travel demand estimation (Liu and Chen, 2017; Wu et al., 2018) and automated dataset collection (Adu-Gyamfi et al., 2017) is already increasing.

From the machine learning perspective, an activity scheduler is a *generative model* (Bishop, 2006), which means that it represents a probability distribution among possible activity schedules, from which new schedules can be sampled.[2] This probability distribution can be conditioned on various variables, such as those describing the modelled person, their household, or seasonal effects.

In this paper, we make the following contributions:

1. We introduce a concept of Data-Driven Activity Scheduler (DDAS) and its implementation. The architecture of DDAS is significantly simpler than the architecture of rule-based activity schedulers.
2. We demonstrate the DDAS concept using our own proof-of-concept implementation and provide a variety of technical details in order to help readers implement their own DDAS-powered models.
3. We propose a novel Activity Attractor Model (AAM) which is a DDAS module responsible for selecting exact locations (specific attractors – not regions) where the activities take place. This seems to be the first *location choice* model of this spatial granularity which can be immediately deployed to a different area than the area it was trained on. One practical implication of this property is that new attractors (e.g, new shops or restaurants) can be added without retraining the model.
4. We compare the validity of DDAS to the validity of a rule-based AgentPolis Scheduler (Čertický et al., 2015). For this, we use the Validation Framework for Activity-Based Models (VALFRAM) (Drchal et al., 2016).

The paper is organized as follows: In Section 2, we put the paper in historical context and provide an overview of the state of the art. Section 3 describes data used for training and validation of the activity scheduler. In the following Section 4, we introduce the DDAS. Section 5 deals with scheduler module training and overall DDAS validation. Finally, Section 6 concludes the paper.

2. Preliminaries

To understand the motivations behind data-driven activity-based modelling, it is useful to see this approach in a broader context of transport modelling and know the history of its emergence.

2.1. Transport modelling overview

Since the 1970s, we have seen numerous attempts to study transport systems by *mathematical methods* and *analytical modelling*, as

[1] https://www.google.com/maps/timeline.

[2] More common *discriminative* models (Bishop, 2006), such as the well-known linear regression, produce point estimates instead of complete distributions.

exemplified in (Geoffrion, 1976) or (Hall, 1986) and overviewed in (Langevin et al., 1996; de Dios Ortuzar and Willumsen, 2011). However, analytic models were often too abstract for expressing relevant aspects and dynamics within transport systems, which lead to the new paradigm of *simulation modelling* (Wilson et al., 1969). Simulations have since then been extensively utilized in the transport research, as a powerful tool for the analysis of system's behaviour.

Typically, the simulated system's behaviour was governed in a centralized top-down manner. Self-initiated interactions and communication among individual actors were impossible. To overcome these limitations, the *agent-based (simulation) modelling* was introduced. It was especially useful for studying complex self-organizing systems (Klügl, 2009). These models are based on the concept of autonomous entities termed *agents*, situated in a shared environment which they perceive and act upon, in order to achieve their own goals. Good examples of agent-based models are AgentPolis (Čertickỳ et al., 2015), SUMO (Behrisch et al., 2011) or TRANSIMS (Smith et al., 1995).

In this paper, we focus on a specific kind of agent-based models, called *activity-based models* (Ben-Akivai et al., 1996), in which the agents plan and execute so-called *activity schedules* – finite sequences of *activity instances* interconnected by *trips*. Each activity instance has a specific *type* (e.g., `work`, `school` or `shop`), *location*, desired *start time* and *duration*. Trips between the activity instances are specified by their transport mode (e.g., `car` or `public transport`). We refer readers to the recent review of the *activity-based models* in (Rasouli and Timmermans, 2014a), here we give only a short overview of methods related to our approach.

An early work on the topic is represented by the CARLA model, developed at Oxford (Jones et al., 1983), followed by STARCHILD, which was often referred to as the first operational activity-based model (McNally, 1986). Models like SCHEDULER (Gärling et al., 1994), TRANSIMS (Smith et al., 1995), CEMDAP (Bhat et al., 2004), SacSim (Bowman et al., 2006), actiTopp (Hilgert et al., 2017), the two-level dynamic model (Dianat et al., 2017) or the walk-bicycle ABM (Aziz et al., 2018) are just a few more recent examples of the activity-based modelling approach.

Even more relevant in the context of this paper is the ALBATROSS model (Arentze and Timmermans, 2000) implemented in 2000, featuring an activity scheduling process which utilized machine learning methods. Three years later, the Toronto Area Scheduling model for Household Agents (TASHA) (Miller and Roorda, 2003) was developed. Similarly to our DDAS scheduler, it generated activity schedules and travel patterns for a twenty-four-hour period representing a typical weekday. Unlike DDAS, TASHA used heuristic rule-based methods to organize and schedule activities of individual members of modelled households. A more recent example of a model featuring a sophisticated activity scheduling algorithm is ADAPTS (Auld and Mohammadian, 2012) implemented in 2012 to model selected counties of the Chicago region. The research behind ADAPTS focused on the ordering and timing of individual planning decisions.

Activity-based modelling with a custom scheduling algorithm was applied in Europe in 2015 when a Fully Agent-based Simulation Model of Multimodal Mobility in European Cities (Čertickỳ et al., 2015) was developed using the AgentPolis modelling framework and validated on Prague and South-Moravian Region in Czech Republic. The ordering of agents' decisions was inspired by ALBATROSS and the decisions themselves were implemented using both the machine learning methods and expert knowledge.

All these models rely on rules and heuristics generated by transport behaviour experts to some extent.

2.2. Data-driven activity-based models

The increasing availability of mobility-related data can reduce our reliance on expert knowledge. Some of the hand-crafted rules and heuristics in activity-based models can be replaced by machine learning (ML) methods, such as neural networks, decision trees, etc. ML methods can be used to implement individual decisions of the agents based on a training dataset, which is representative of the behaviour of the modelled population. These datasets are typically questionnaires, surveys or origin-destination matrices.

This data-driven approach to activity-based modelling offers some potential advantages:

1. Complex processes, such as cognitive functions and human behaviour, are usually hard to tackle using exact analytical or mathematical methods. Machine learning is often used to approximate such complex functions with sufficient precision, even without the understanding of underlying psychological or biological aspects.
2. Activity-based models are typically focused on a population of a specific area. The data-driven approach makes it easier to adjust the models for new locations. ML models just need to be trained again, using datasets gathered in the target area.
3. From a technical perspective, it tends to be easier to implement a data-driven model, and even a small team should be able to do it. The involvement of psychologists, sociologists or other domain experts is not necessary in this case.

One potential disadvantage of data-driven models lies in the limited explainability of machine learning modules involved. However, recently developed methods, such as (Turner, 2016; Samek et al., 2017), should significantly mitigate the problem.

The activity schedulers of the models mentioned above are data-driven to a varying degree – the ratio of ML-based and rule-based algorithms involved in their design differs. The ML-based modules used in the schedulers are often implemented as a simple logit, probit or logistic and linear regression models (Miller and Roorda, 2003; Bhat et al., 2004; Bowman et al., 2006; Auld and Mohammadian, 2012; Hilgert et al., 2017). More powerful decision and regression trees are also utilized, in some examples (Arentze and Timmermans, 2000; Smith et al., 1995; Auld and Mohammadian, 2012). Tree ensembles such as random forests are employed in (Rasouli and Timmermans, 2014b; Ghasri et al., 2017; Hafezi et al., 2018). Generative artificial neural network models are employed in (Čertickỳ et al., 2015) and Bayesian networks in (Janssens et al., 2004). The ML methods are most often used to select the activity properties (e.g., activity type, duration, start time, location), trip properties (e.g., mode) or schedule properties (e.g., schedule template).

Unlike the schedulers mentioned above, the data-driven schedulers, such as DDAS, can rely on statistical machine learning instead of trying to replicate the structure of human's cognitive processes. DDAS solves the problem of activity scheduling using a *generative model* of the probability distribution. In order to produce the schedules for the agents, the model is randomly sampled. The need for expert-designed rules and heuristics imposing spatial or temporal constraints on agent's behaviour has been greatly reduced. The reasoning is that with enough data, these constraints can be automatically learned by ML methods. The actual planning process is also trained based on examples.

Complex decompositions of the activity planning process in the schedulers above were mainly motivated by the knowledge about human cognitive processes. In contrast to this, the structure of more data-driven schedulers (such as DDAS – see Section 4) can be determined by practical and technical issues. This potentially leads to simpler systems that are easily described and implemented, which in turn increases the reproducibility of the results. Also, an accurate data-driven model can help in revealing behavioural principles behind actual human decision making.

3. Data

In this section, we describe the types of data required for our Data-Driven Activity Scheduler (DDAS). Various data sources are used in three different ways: as the inputs for the decisions during runtime, to train individual decision modules of the scheduler using ML algorithms and for model validation.

3.1. Transport network and path planners

Transport network representation (roads, railways, etc.) and path planners are necessary for the agents because they need to plan their activities and trips in the city while taking into account the spatial context and restrictions. The physical part of the transport network is based mainly on the Open Street Maps (OSM) – a publicly accessible map files[3] released under Open Data Commons Open Database Licence (ODbL). OSM uses a topological data structure with four core elements: Nodes, Ways, Relations and Tags. It is usually distributed in Extensible Markup Language (XML)[4] format. Non-physical properties of the transport system, such as transport restrictions (speed limits, mode permissions, etc.), are also extracted from OSM.

In addition to the detailed model of the physical world, we also maintain the representation of all the local **administrative areas**: relatively small, mutually exclusive polygons covering the whole modelled region. These provide an additional (lower) level of detail for situations where the data is only available with lower spatial granularity or for the operations, where full spatial detail is not needed.

A detailed **public transport** (PT) model is also needed so that agents can plan their trips. Our PT model contains public transport stations, routes, timetables and individual vehicle trips specifying sequences of stations and arrival/departure times. All this is loaded from General Transit Feed Specification (GTFS)[5] files. GTFS feed usually contains several CSV-like (Comma Separated Value) text files. Each file models a particular aspect of transit information: stops, routes, trips, and other schedule data.

3.2. Attractors

The scheduler also needs access to attractors, sometimes called points of interest (POIs). They represent all the places, where an agent can perform activities. Attractor in our model is a data structure holding the following information about each location:

- latitude-longitude coordinates,
- types of activities that can be executed at a given attractor,
- opening hours of the attractor,
- additional information such as attractor's name, its visit count and attractivity, etc.

Attractors represent all the workplaces, schools, residential locations, leisure time venues like restaurants, cinemas, tourist attractions, etc. We collect the attractors from three sources:

1. We extract the initial set of attractors from OSM maps. We look for OSM tags with specific values of keys like *amenity, land use, building, tourism, shop*, etc. For example, OSM tags that have amenity = school are used to create attractors allowing the "school" activity or tags with *shop* = *supermarket* are used for "shopping" attractors.
2. We collect additional attractors by making numerous HTTP requests to public Foursquare API. Foursquare API provides a large database of attractors (these are called "Venues" in Foursquare terminology). Each attractor comes with a collection of metadata including its location, opening hours (in some cases), numbers of daily check-ins by users of Foursquare mobile application[6] and a

[3] http://openstreetmap.org/.
[4] http://www.w3.org/XML/.
[5] http://developers.google.com/transit/gtfs/reference.
[6] Foursquare check-ins correlate with the attractivity of POI, but one needs to be careful when using them. Keep in mind that the demographic composition of Foursquare users does not necessarily correspond to the whole modelled population. They can also be somewhat useful during model

category, we use to assign allowed activity types. New attractors are merged with the ones from the previous step, based on their location and name.
3. Finally, we use Google Maps API to collect missing information about collected attractors – especially the opening hours. This step is relatively slow due to Google Maps API restrictions.

The idea to mine attractor data from social media, including Twitter, Foursquare (Kuflik et al., 2017; Chaniotakis et al., 2017) and Yelp (Bou Mjahed et al., 2017), has been recently suggested by several researchers. A good overview of this topic, including the discussion on possible social biases, can be found in Rashidi et al. (2017).

3.3. Socio-demography

Additional types of data are needed to generate the population of agents with their demographic attributes (age, gender, education, marital status, financial income, economic activity, driver's licence, etc.), assign them into shared households and set their place of employment or school.

Data sources for this purpose are specific to each scenario. For our proof-of-concept implementation, we used the census data set collected by the Czech Statistical Office. Census data tend to be a good source of socio-demographic attributes in many scenarios since they are usually periodically collected by the local administration and publicly accessible. They usually come with varying levels of aggregation and spatial granularity and can be combined to generate a realistic synthetic population. In some cases, origin-destination matrices (for example from mobile phone tracking datasets) can be used here - especially to make the workplace/school assignment more precise.

Population synthesis is an interesting problem in itself, but the details about how we solve it are outside the scope of this paper. For now, it is only important to note that DDAS scheduler requires a population of agents with demographic attributes and their place of residence and work/school.

3.4. Travel diaries

Travel diaries provide fundamental data about the travel behaviour of the modelled population and therefore can be used to train the ML algorithms behind agent's decision making.

For our implementation, we used the travel diary data set collected by Transport Research Center in Czech Republic. It covered the travel-related behaviour of 2600 participants from 1000 households during the time period of up to four days. The activities within the dataset were characterized by their type (`sleep`, `work`, `school`, `leisure`, `shop`), location, start time and end time. Trips had a specified origin and destination location, transport mode (`walk`, `car`, `bike`, `pt`) and start/end times. The diaries also contained anonymized demographic information about the participants. In addition to training of ML algorithms, we also used some of this data for the model validation.

4. Scheduler description

In this section, we describe the Data-Driven Activity Scheduler (DDAS) in detail. First, we formalize the activity scheduling as a machine learning problem and discuss the related difficulties. Then, we propose the solution that involves separating DDAS into four different modules: Activity Type Model (ATM), Activity Duration Model (ADM), Activity Attractor Model (AAM) and Mode Choice Model (MCM). We describe each of the core modules in detail. Finally, at the end of the section, we introduce an algorithm that actually generates the schedules using the modules.

The approach represented by DDAS is designed to be applicable to activity-based modelling in general and does not depend on specific location or implementation. However, we often include specific implementation-related details throughout the following text, since they might prove useful for readers trying to implement their own DDAS-based models.

4.1. Design considerations

We are interested in sampling schedules s from the conditional distribution $p(s|k)$, where k represents any prior knowledge such as the socio-demography of the agent for which the schedule is generated. From a machine learning perspective, our task can be defined as finding a *generative model* (Bishop, 2006) $p_\theta(s|k)$, where θ represents trainable parameters of the model.

An activity schedule can be formalized as a sequence of activities $s = (a_1, a_2, ...)$ with each activity a_i decomposed as $a_i = (t_i, s_i, d_i, l_i, m_i, d_i^T)$, where t_i is the activity type, s_i its start time, d_i its duration, l_i the associated attractor (location), m_i the main mode of transport of the trip preceding the activity and d_i^T is the trip duration.

The distribution $p_\theta(s|k)$ can then be factorized as:

(*footnote continued*)
validation, but mostly just to detect the obvious problems with specific attractors. Other data sources can be used to assign the attractivity of POIs, but unlike Foursquare check-ins, they tend to be expensive.

$$p_\theta(s|k) = p_\theta(a_1, a_2, ...|k) = p_\theta(a_1|k)\cdot p_\theta(a_2|a_1, k)\cdot p_\theta(a_3|a_1, a_2, k)\cdot ..., \tag{1}$$

which allows us to schedule the activities sequentially. The i-th activity is generated using all the preceding activities using transition $p_\theta(a_i|\boldsymbol{a}_{i-1}, k)$ which is impractical as their number grows infinitely.[7] To overcome this problem, the conditional probability is often simplified to $p_\theta(a_i|h(\boldsymbol{a}_{i-1}), k)$, where $h(\boldsymbol{a}_{i-1})$ is typically a function of a subset of $\boldsymbol{a}_{i-1}$ only. As an example, when $h(\boldsymbol{a}_i) = a_i$, we talk about the first-order Markov chain (Bishop, 2006), where the current activity depends only on the previous one.

When using certain machine learning methods, such as recurrent neural networks (Goodfellow et al., 2016), the function h itself can be a subject to learning – it can be represented by the network's *hidden state*. Recurrent neural network architectures such as the well-established LSTM are known to be able to work with histories of thousands of steps (Goodfellow et al., 2016), which would be more than enough for our activity schedules. Even though the use of recurrent neural networks might be tempting, as we could potentially end up with a single end-to-end trained fully-differentiable model, we need to cooperate with external non-differentiable algorithmic systems, such as path planners, which makes this approach complicated. Possible solutions to these interfacing problems might involve: (1) making the non-differentiable software modules differentiable, (2) using differentiable approximations of the modules during the training phase or (3) trade gradient-based learning methods for meta-heuristics such as evolutionary algorithms (Michalewicz, 1996).

The approach we have chosen for DDAS is a compromise between the approaches mentioned above. DDAS uses a custom hidden state function h which gives access to the whole history of recent activities, formally $h(\boldsymbol{a}_i) = (a_i, \boldsymbol{c}_i)$, where $\boldsymbol{c}_i$ encodes an aggregated history of $a_1, ...a_{i-1}$ by a fixed-length vector (see details in the following sections). Moreover, the transition probability is further factorized:

$$\begin{aligned} p(a|h(\boldsymbol{a}')) &= p(t, d, l, m, d^T|h(\boldsymbol{a}')) \\ &= \underbrace{p(t|h(\boldsymbol{a}'))}_{\text{ATM}}\cdot\underbrace{p(d|t, h(\boldsymbol{a}'))}_{\text{ADM}}\cdot\underbrace{p(l|t, d, h(\boldsymbol{a}'))}_{\text{AAM}}\cdot\underbrace{p(m, d^T|t, d, l, h(\boldsymbol{a}'))}_{\text{MCM}}, \end{aligned} \tag{2}$$

where θ, dependence on k and subscripts were omitted for better readability. In the formalism above, the prime symbols denote previous time steps (i.e. $a = a_i$, while $\boldsymbol{c}' = \boldsymbol{c}_{i-1}$). The acronyms below the factors indicate correspondence to individual DDAS modules. The activity start time s is omitted, since it can be simply computed as $s = s' + d' + d^T$.

The reasoning behind the factorization is:

- Separate training is particularly useful here since the task is to find a generative model. The variables produced by the model need to be both discrete (e.g., the activity type t) and continuous (activity duration d). If we wanted to train a single model to produce both of these kinds of outputs, our choices of machine learning algorithms would be severely limited.
- The separation allows easier cooperation with external software, such as path planners. Our Activity Attractor Model (AAM) is a particular example of this.
- Factorized modules can be trained separately which allows us to choose a different machine learning paradigm for each and maintain a better insight into the training process.
- Available training data is often incomplete and the decomposition of the model into smaller modules can help deal with that. For example, travel diary datasets often have a limited spatial precision (activities are assigned to encompassing areas instead of specific attractors). In order to deal with this problem, in the Activity Attractor Model we combine travel diary data with separate attractor visits data (see Section 3.2). This requires a specialized approach to training which would be inappropriate for the tasks solved by the other modules.

Note that similar probability factorization was already presented by Kitamura (Kitamura, 1983). In this work, only the activity types t_i were considered. The spatiotemporal attributes were covered later in (Kitamura et al., 1997). The theoretical analysis, however, deals with the full history $\boldsymbol{a}_{i-1}$ only and authors do not discuss the consequences regarding the training of generative models.

The sequential approach we use in DDAS might seem to be in contradiction with the results of previous research, such as (Doherty, 2000), which suggests that individuals are more likely to schedule based on priorities than in purely sequential fashion. It is, however, important to understand that the sequentiality, i.e, the factorization (Eq. (1)), in DDAS is rather a tool which facilitates to generate schedules of a variable length using fixed-size input machine learning models. The prioritization and corresponding decision making can be encoded in the structure and parameters of the models involved. More detailed discussion on this topic can be found in Kitamura (1983).

4.2. DDAS architecture

As discussed in the previous section, DDAS is composed of four core modules, each responsible for one kind of agent's decisions. Each of the four decision modules (ATM, ADM, AAM and MCM) is a generative supervised machine learning model. We describe their objectives with respect to the scheduler, enumerate the data on their input and output and explain how they are trained.

DDAS uses a sequential approach to decision making which corresponds to the factorization presented in Eq. (2). The process is

[7] We use the notation $\boldsymbol{a}_i = (a_1, ..., a_i)$.

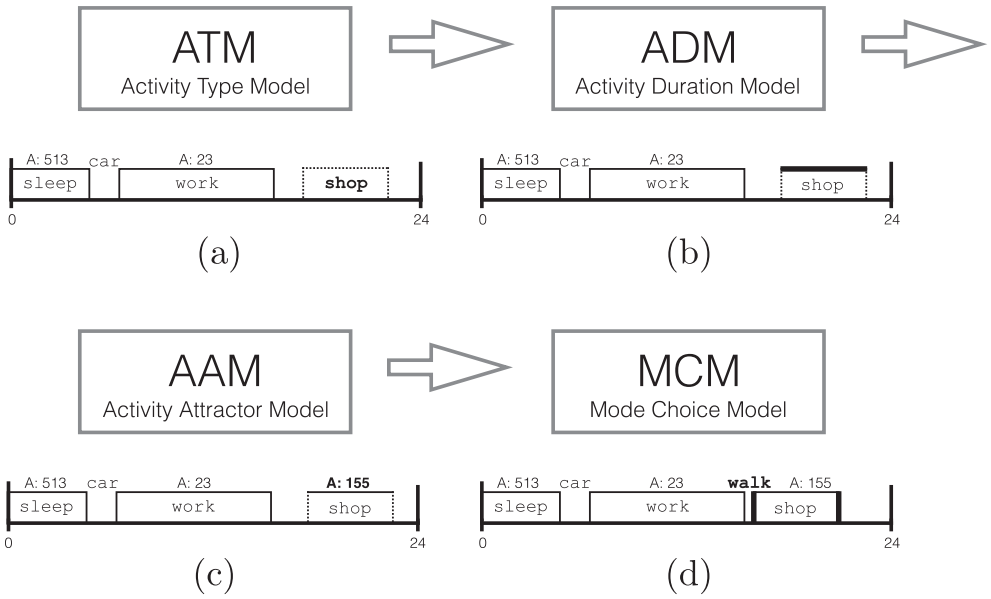


Fig. 2. Sequential process of adding an activity instance to the schedule.

illustrated in Fig. 2. When adding a new activity instance to the schedule, DDAS uses these four decision modules in the following order:

1. **ATM:** Activity Type Model selects the type of the activity that is being added to the schedule. It outputs a probability for each possible activity type including a specialized "`none`" type which marks start and end of the schedule. The probabilities determine a probability mass function from which we sample $p(t|h(\boldsymbol{a}'))$ activity type instances during schedule generation. We use a decision tree classifier in the current version of DDAS, although other ML algorithms might be used as well.
2. **ADM:** Activity Duration Model's purpose is to generate a duration for the activity of the type selected by ATM. In this case, a model of a continuous probability distribution $p(d|t, h(\boldsymbol{a}'))$ is needed. We use a probability density function model based on a regression tree. See details in Section 4.4.
3. **AAM:** Activity Attractor Model outputs $p(l|t, d, h(\boldsymbol{a}'))$ a discrete distribution over all the attractors available for a given activity instance. We cast this task as a multi-objective optimization problem. Based on the attractor visit count (mentioned in Section 3.2) and the expected duration of the trip leading to the activity, the objectives are: 1) minimize the difference between measured and modelled visit count and 2) minimize the trip durations. Details are given in Section 4.5.
4. **MCM:** Mode Choice Model finally selects a mode of transport for agent's trip to given activity instance (which in turn affects the trip's duration). Since the mode choice can be expressed as a discrete probability distribution $p(m, d^T|t, d, l, h(\boldsymbol{a}'))$, we decided to use a decision tree just as in case of ATM. Note that the trip duration d^T is directly computed using the path planner when the mode is selected, in other words, we train a generative model $p(m|t, d, l, h(\boldsymbol{a}'))$ and d^T is computed deterministically afterwards. See Section 4.6 for details.

The scheduler sequentially adds new activities until full day schedule is generated. Fig. 2 shows an example of adding a new activity to a partially generated schedule composed of `sleep` and `work` activities. First, (a) the ATM generates the type of the following activity (`shop`). Note that neither spatial nor temporal properties of the activity are known (indicated by the dotted lines). Subsequently, (b) the ADM samples duration of the activity (notice the bold line). Next, (c) the AAM selects an attractor (attractor ids are depicted above the activity boxes). Finally, (d) the MCM selects a transport mode (`walk`) and estimates the corresponding trip duration. This completely fixes the temporal properties of the activity (start and end times are marked by bold vertical lines).

These four models require several types of input data (denoted by k above). Each model uses the same set of *common features* accompanied by a different, module-specific set of *context features* which depend on the outputs of the previous models. *Common features* currently consist of the following three sets:

- **socio-demography** (denoted `soc` in this paper) is based on data described in Section 3. We specifically use the following demographic features of agents: `household size`, `age`, `gender`, `car available in the household`, `student`, `education achieved` (`low`, `mid` or `high` level), `driver's license` and `public transport pass`.
- **reach descriptor** (denoted `reach`) expresses an approximate duration of trips from agent's home to the place of his main activity

(`work` or `school`) using each mode of transport.[8] The `reach` feature is an estimation – we compute it on the level of AA (administrative areas) and only for one time in a day (8:00 AM). Reach descriptor might be understood as a spatiotemporal subset of socio-demography, as it is based on fixed locations of the home and the main activity and their mutual accessibility. Nevertheless, we decided to treat it separately as it is computed rather than directly taken from the synthesized population.

- **activity type count** (denoted `count`) keeps a separate counter for each activity type. Each counter gives a number of occurrences of the corresponding activity in the agent's schedule so far. This is available during the scheduling process for all the activity types with the exception of `none` type. The `count` feature represents an aggregated history $\boldsymbol{c}$ as discussed in Section 4.1.

The model-specific *context features* are described in corresponding sections below.

4.3. Activity Type Model (ATM)

Activity Type Model (ATM) outputs a probability distribution over the *types* for the *next activity*, based on the *common* and the *context features*. The *context features* are: *type of the current activity* and the *end time of the current activity*.[9]

In our implementation, we consider activity types `sleep`, `work`, `school`, `leisure`, `shop` and `none`. We implement ATM as a decision tree[10] since it produced the best results on our dataset. Other paradigms, such as random forest (Breiman, 2001) or Multi-Layered Perceptron (MLP) with softmax output layer (Bishop, 2006), are also viable options.

An example of an ATM decision tree is shown in Fig. 3. The depth of this particular tree was limited to three for clarity. When evaluating the decision tree, we follow a path starting in the root node, making decisions whether to continue to the left or to the right subtree based on a *split point* represented by the node and a value of the corresponding feature. For *categorical features* such as *current activity type*, a node can, for example, check whether the *current activity type* (denoted t' in the figure) is equal to `sleep`, if so the evaluation will continue to the left subtree or to the right one otherwise. Similarly, for a *continuous feature* such as *current activity end* (denoted e'), the decision is made based on its value being less or equal to a selected threshold defined by the node's *split point*. Each decision tree node hence represents a subset of the original training data (see the number of samples in each node in the figure). In the discriminative setting, the tree response is determined by descending to a leaf and taking a majority over its training subset[11] class labels. In the generative setting, which is our case, the class membership probabilities are defined as class proportions w.r.t. the subset. Note that a similar approach to obtain class probabilities was used in the ALBATROSS model (Arentze and Timmermans, 2000).

4.4. Activity Duration Model (ADM)

After ATM samples the activity type, the Activity Duration Model (ADM) determines the activity duration (denoted as the *next activity duration* below). Unlike ATM, which produces a discrete probability distribution (i.e., categorical distribution), the ADM models a continuous one.

Note that an alternative scheduler ALBATROSS modelled all the continuous distributions as discrete. Such discretization tends to harm the model performance due to information loss, while on the other hand, it can make the training easier (Kotsiantis and Kanellopoulos, 2006).

As the input, ADM uses *common features* as well as *context features*: *type of next activity* (sampled by ATM) and the *end time of the current activity*.

ADM is implemented as a regression tree. Regression trees work similarly to the decision trees described above. Their response in the discriminative setting is typically defined as an average of continuous labels over the training subset samples represented by the *leaf subset*. There are multiple options for a generative regression tree models. One would be to model the *leaf subset* using a parametric distribution (such as normal distribution) the parameters of which would be determined by maximum likelihood estimation or some other similar method. The second approach is to treat the *leaf subset* itself as a definition of an empirical distribution. Then, taking samples from this distribution can be simply realized by uniform sampling over the *leaf subset* labels with replacement such as in Ghasri et al. (2017). We use this latter, simpler, method for our ADM implementation.

4.5. Activity Attractor Model (AAM)

After the activity has its type determined by ATM and duration by ADM, the Activity Attractor Model (AAM) is used to select a specific attractor for it. This positions the activity in space. We train a separate AAM model for each type of *flexible activity*: `leisure` and `shop`. Formally, AAM can be defined as a function:

[8] If there is no connection from agent's home to his main activity location using a given mode, the `reach` feature holds a value of -1. Otherwise, its value is computed as an average of estimated durations between 3 randomly selected locations per AA.

[9] Note that, in contrast with the general description in Section 4.1, we only use a subset of features provided by $h(\boldsymbol{a}')$. Also, for the sake of simplicity, we use the end of the current activity e' instead of its start and duration, where $e' = s' + d'$.

[10] There exist multiple varieties of decision and regression trees. In this work, we used solely those of Scikit-learn machine learning library (Pedregosa et al., 2011), which implements binary decision and regression trees.

[11] Denoted as the *leaf subset* in the following.

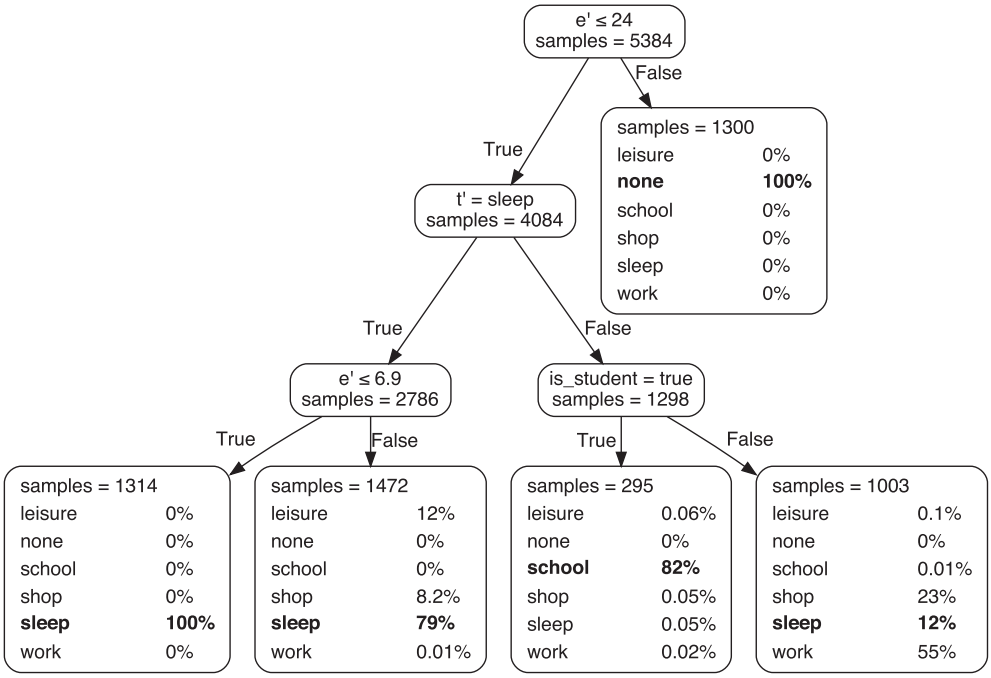


Fig. 3. ATM decision tree example. Each inner node represents a decision which might be based either on a discrete (here *current activity type* denoted by t' and `is_student`) or a continuous (*current activity end* denoted by e') feature and a corresponding *split point*. Each node represents a subset of training data samples, where for leaves the fractions of the original data corresponding to different target classes (*next activity type*) are shown. These fractions over the *leaf subset* define a categorical distribution from which we can sample (the activity types having highest probability are marked bold).

$$\boldsymbol{y} = AAM(\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{x} \in X$ is a vector of input features describing so-called *attractor demand* and $\boldsymbol{y} \in Y$ is a model's response – a vector defining a categorical distribution over all the attractors $j \in A$ of selected flexible activity type. Therefore, $\sum_{j \in A} y_j = 1$ and $y_j \in [0, 1]$.

We assume that the *attractor demand*, represented by the feature vector $\boldsymbol{x}$ provides enough information for the probabilities y_j to be inferred. It should include at least some information about: (1) spatiotemporal relationships between the current schedule state and all available attractors $j \in A$, (2) *attractiveness* a_j defined for each attractor $j \in A$, and (3) agent's socio-demography, possibly including the `reach` descriptor. An example of a spatiotemporal relationship might be a distance between current activity attractor $i \in A$ and a candidate attractor for the next activity $j \in A$. Such distance can be defined in more than one way – we use an estimated duration of the trip between the two locations.

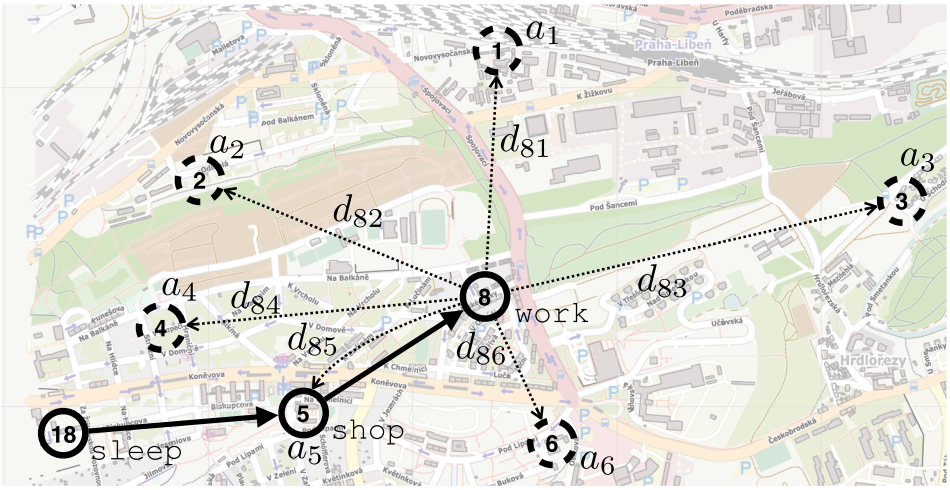


Fig. 4. Demand for the `shop` attractors. Each circle marks an attractor and the number inside is its identifier. Labels $d_{81}, \ldots, d_{86}$ are distances from the attractor 8 and $a_1, \ldots, a_6$ denote the *attractiveness* of corresponding attractors.

Fig. 4 shows an example of *attractor demand* for a new `shop` activity. After `sleep`, `shop` and `work` activities were scheduled, another `shop` type activity is requested (its type was selected by ATM and its duration generated by ADM). All the `shop` attractors are shown ($A = \{1, ..., 6\}$). In this case, the corresponding feature vector may be constructed as $\boldsymbol{x} = (d_{81}, ..., d_{86}, a_1, ..., a_6, \text{soc}, \text{reach}, \text{count})$, where $d_{81}, ..., d_{86}$ are distances between the current attractor with ID 8 and all the available `shop` attractors in consideration. Symbols $a_1, ..., a_6$ denote the value of *attractiveness* of the attractors.

The *attractiveness* of an attractor is harder to explicitly define and objective data is not easy to come by. Nevertheless, when selecting attractors, people tend to favour those that are better for some reason (higher *attractiveness*) and minimize their travel time or distance. Instead of an actual objective measure of attractiveness, we use a proxy feature: the number of visits $\hat{\boldsymbol{v}}$, which can be collected for example from Foursquare API, as described in Section 3.2. In our case, each element $\hat{v}_j, j \in A$ expresses an average number of visits per day. Even despite the fact that we use such derived data, the AAM should be able to infer the actual *attractiveness* internally.

Based on $\hat{\boldsymbol{v}}$, we can further define a vector of normalized attractor visits $\boldsymbol{v} = \hat{\boldsymbol{v}}/|\hat{\boldsymbol{v}}|$ giving a categorical distribution, hence, $\sum_{j \in A} v_j = 1$ and $v_j \in [0, 1]$. For an optimal (or ground truth) activity attractor model AAM^* we have:

$$\boldsymbol{v} = \mathbb{E}_{x \sim D}[AAM^*(\boldsymbol{x})], \tag{4}$$

where D is a distribution over *attractor demands* described by the feature vectors.[12]

In a typical supervised learning setup, we would gather a training dataset: $\mathcal{T} = \{(\boldsymbol{x}^{(k)}, t^{(k)}) \in (X \times A) | k \in 1, ..., m\}$, where the label $t^{(k)} \in A$ denotes the attractor visited for the k-th demand. Then, we would train a model to output the class probabilities for each attractor (as it was done for activity types in case of ATM). This is the approach that was taken by the ALBATROSS scheduler. However, in that case, the whole encompassing areas were considered (instead of the individual attractors) and their number was fixed (Arentze and Timmermans, 2000).

The problem with the approach described above is that training set $\mathcal{T}$ has to be very large in order to represent the underlying distribution. A trivial, optimistic lower bound is $m = |A|$ when each attractor is visited at least once (at least one sample for each attractor). To address this issue, we suggest approximating the original problem of *AAM* training by solving a related multi-objective optimization problem, with the following objectives:

1. Minimize the difference between normalized visits generated by *AAM* and actual normalized visits from the data, i.e., to search for optimal set of *AAM* parameters:

$$\theta^* = \text{argmin}_\theta ||\boldsymbol{v} - \mathbb{E}_{x \sim D}[AAM_\theta(\boldsymbol{x})]||, \tag{5}$$

2. Minimize an expected value of distances travelled to the attractors being sampled from a distribution given by $AAM_\theta(\boldsymbol{x})$:

$$\theta^* = \text{argmin}_\theta \mathbb{E}[\text{dist}(AAM_\theta(\boldsymbol{x}))], \tag{6}$$

where $\text{dist}(\boldsymbol{y})$ function randomly samples an attractor j based on a distribution given by its argument $\boldsymbol{y}$ and returns a distance between the current attractor i and the next attractor j.

Other possible issue stems from the fact that the number of elements of x is proportional to the number of attractors $|A|$ which can be impractically high – high input dimension implies a model of high complexity (e.g., number of parameters) which will more likely overfit. To mitigate this problem, we further propose rearranging the training set to:

$$\mathcal{T} = (\{\boldsymbol{x}_j^{(i)} | i \in 1, ..., m; j \in A\}, \boldsymbol{v}), \tag{7}$$

where for demand i, the $\boldsymbol{x}_j^{(i)}$ is a subset of features of $\boldsymbol{x}^{(i)} \in X$ related to the attractor j. In this paper we consider it to be the concatenation:

$$\boldsymbol{x}_j^{(i)} = (d_{ij}, \hat{v}_j, \boldsymbol{o}_i), \tag{8}$$

where d_{ij} is an estimated trip duration between current attractor *demand* i and the attractor j of the next activity,[13] $\hat{v}_j$ is the number of visits of j per day (a proxy for a_j) and $\boldsymbol{o}_i$ concatenates all *other* features in consideration such as `soc`, `reach` or `count`. Note that, unlike in standard supervised-learning setup, where the training data is a set of pairs of input features $\boldsymbol{x}$ and labels t, here we have a set of input features describing *attractor demands* and a single label $\boldsymbol{v}$.

The AAM_θ model based on $\mathcal{T}$ is defined as follows:

1. First, the utility $u_j^{(i)} = f_\theta(\boldsymbol{x}_j^{(i)})$ is computed for each $j \in A$, where f_θ represents a regression model parameterized by θ. In our implementation, the model is realized by MultiLayer Perceptron with linear neurons in the output layer.
2. Second, softmax (Bishop, 2006) function is applied to the utilities:

[12] Here, we expect that demand feature vectors contain all the information needed to infer the correct vector $\boldsymbol{y} = AAM^*(\boldsymbol{x})$.

[13] We use minimum estimated trip duration considering all available modes of transport.

$$y_j^{(i)} = \frac{\exp(u_j^{(i)})}{\sum_{k \in A} \exp(u_k^{(i)})}. \tag{9}$$

Now, each $\boldsymbol{y}^{(i)} = AAM_\theta(\boldsymbol{x}^{(i)}) = (y_1^{(i)}, ..., y_{|A|}^{(i)})$ represents a categorical probability distribution, where element $y_j^{(i)}$ can be interpreted as a probability of choosing attractor j for *demand* i.

We can compute an estimation of the expected value of normalized visits $\mathbb{E}_{x \sim D}[AAM_\theta(\boldsymbol{x})]$ for each attractor j considering all demands as:

$$y_j = \frac{1}{m} \sum_i y_j^{(i)}. \tag{10}$$

The task is to find parameters θ of f_θ which minimize differences between actual normalized visits $\boldsymbol{v} = (v_1, ..., v_{|A|})$ and the modelled ones $\boldsymbol{y} = (y_1, ..., y_{|A|})$. Hence, we define a *visit loss function* $\mathcal{L}_V$ according to Eq. (5) as the following mean squared error:

$$\mathcal{L}_V = \frac{1}{|A|} \sum_j (v_j - y_j)^2. \tag{11}$$

Similarly, we define the *distance loss function* $\mathcal{L}_D$ for the second objective specified by Eq. (6) as:

$$\mathcal{L}_D = \frac{1}{m|A|} \sum_i \sum_j y_{ij} d_{ij}, \tag{12}$$

which corresponds to an estimation of the expected value of distance travelled for all demands considering all attractors based on model outputs.

The compound loss function is based on a linear combination of the two objectives:

$$\mathcal{L} = \alpha \mathcal{L}_D + (1 - \alpha) \mathcal{L}_V, \tag{13}$$

where $\alpha \in [0, 1]$ balances both $\mathcal{L}_D$ and $\mathcal{L}_V$. The loss function is differentiable with respect to the parameters θ, therefore any standard optimization method such as Stochastic Gradient Descent (Bishop, 2006) can be used for the optimization.

The selection of a final model is not straight-forward due to the multi-objective nature of the problem. In practice, we execute multiple runs for several values α, collecting both $\mathcal{L}_D$ and $\mathcal{L}_V$. Then, we select a set of non-dominated AAM models (those lying on a Pareto front) evaluating them in the context of the whole scheduler and finally choosing a single model giving the best overall results. See Section 5.4 for the details.

Training AAM using a large number of demands and attractors might be computationally expensive (since $|\mathcal{T}| = m|A|$ elements).

Fig. 5. Modelled area of South Moravian Region of Czech Republic. The red rectangle shows the area that was used to train AAM. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The fact that the number of $x_j^{(i)}$ elements is fixed and does not depend on a number of attractors, however, allows us to train on a selected subarea only. Red rectangle in Fig. 5 in the next section shows a subarea we have used to train our AAM models. See Section 5.4 for more details.

4.6. Mode Choice Model (MCM)

The task of Mode Choice Model (MCM) is to select an appropriate mode of transport for the trip from the current to the next activity. Similarly to the ATM, the choice can be described by a categorical distribution and hence we use the same, decision tree based approach. We consider the following four modes: `pt` (public transport), `walk`, `car` and `bike`. The model is conditioned on *common features* as well as on the following set of *context features*:

- *type of the next activity* as used by ADM,
- *type of the current activity* as used by ADM,
- *mode counts* – a vector of the numbers of prior trips in the schedule using a corresponding mode,
- *trip duration estimates* – a vector of estimated trip durations between the current and the next activity (with the location determined by AAM) for each mode.

4.7. Scheduling algorithm

The pseudocode in Algorithm 1 describes how an activity schedule is generated by DDAS. The GENERATE-SCHEDULE() procedure, which is called for each member of the population of agents, sequentially appends new activities to the schedule S.

Algorithm 1. DDAS Schedule Generation

```
GENERATE-SCHEDULE()
 1 S = NEW-LIST()      // list of schedule activities
 2 t' = none           // current activity type
 3 e' = 0              // current activity end
 4 l' = NIL            // current attractor (location)
 5 for i = 1 to MAX-ITERATIONS
 6   c = GET-COMMON-FEATURES()          // common features: soc, reach and count
 7   t = SAMPLE-ATM(t', e', c)          // sleep, work, school, leisure, shop or none
 8   if t = =none                       // the schedule ends with the current activity
 9     break
10   d = SAMPLE-ADM(t, e', c)           // sample duration
11   C = SELECT-ATTRACTORS (t, e', l', c)   // sample attractor candidates
12   m = NIL
13   p = GET-MODE-COUNTS (S)
14   for l ∈ C
15     m, d_t = CHOOSE-MODE (t', e', t, l', l, d, p, c)   // sample mode and trip duration
16     if m is not NIL                  // select first viable attractor
17       break
18   if m is NIL                        // no attractor candidate of C is viable
19     return NIL
20   s = e' + d_t                       // next activity start
21   if s ⩾ 24                          // prevent activity to start the next day
22     return NIL
23   e = s + d                          // next activity end
24   if e ⩾ 24                          // crop activity length
25     e = 24
26     d = 24 − s
27   append NEW-ACTIVITY (t, s, d, d_t, m, l) to S
28   if e = =24                         // the schedule ends with this activity
29     break
30   t', e', l' = t, e, l
31 return S
```

The pseudocode should be self-explanatory, but we add a few remarks:

- The outer loop (line 5) prevents the infinite sequence of activities which might happen due to the randomized nature of the modules.
- The schedule generation can also be terminated by sampling `none` type activity (line 8) or by generating an activity which reaches the end of the day (line 28).
- The SELECT-ATTRACTORS() function samples k (we use $k = 10$ in this paper) attractor candidates denoted C according to the output of

the AAM module. We generate multiple candidates, since there may be no possible connection between the current and next attractor using the mode chosen by the CHOOSE-MODE() function.

- The GET-MODE-COUNTS() function returns a list of trip counts by transport mode, within the schedule S. This is analogous to the `count` extracted by GET-COMMON-FEATURES() function.
- The CHOOSE-MODE() function generates trip durations estimates, checks for attractor's opening hours and samples the modes using MCM.
- The schedule generation may fail if a connection between successive attractors cannot be found (line 18) or if an activity is scheduled to start the next day (line 21). If that happens, we call the GENERATE-SCHEDULE() function up to ten times for the same agent before giving up. These rare failures are caused by occasional errors (e.g., in the transport network) or insufficiencies in the data.

Algorithm 2. DDAS Attractor Selection

```
SELECT-ATTRACTORS (t, e', l', c)
1 if IS-FIXED (t)
2   return FIXED-ATTRACTOR (t)
3 else
4   D = ESTIMATE-DURATIONS-TO-ATTRACTORS (t, l')
5   y = EVALUATE-AAM (t, D, c)
6   return sample k attractors using probabilities y      // k = 10 in our experiments
```

The SELECT-ATTRACTORS() function, described in Algorithm 2, first checks the type of the attractor to be sampled. If the type corresponds to a fixed attractor (`sleep`, `work` or `school` activities) then the attractor was already assigned during the population synthesis and should not be changed. Otherwise (`leisure` and `shop` activity types) we estimate the distances $d_{l'l}$ between the current attractor l' and all allowable target attractors $l \in A$ of type t using ESTIMATE-DURATIONS-TO-ATTRACTORS() function. Here, each estimation is computed as a minimum trip duration between l' and l considering all modes. We use precomputed values for 8:00 AM as getting all the estimations for all the attractors $l \in A$ on the fly would be too computationally demanding. Finally, the EVALUATE-AAM () function selects an AAM corresponding to the activity type t supplying it the estimated distances D and the *common features* c.

Algorithm 3. DDAS Mode Choice

```
1 CHOOSE-MODE (t', e', t, l', l, d, p, c)
2 if t' is none      // initial activity, no need to travel
3   return none, 0
4 D = ESTIMATE-MODE-DURATIONS (l', l, e')
5 D = FILTER-OPENING-HOURS (l, D, e', d)
6 m, d_t = SAMPLE-MCM(l, e', d, D, p, c)
7 return m, d_t
```

Mode choice is covered in Algorithm 3. Based on the current attractor l' and the candidate for the following one l, the path planner is evaluated for each mode using ESTIMATE-MODE-DURATIONS() function. Estimations are returned only for those modes, for which the planner succeeds in finding the trip. The planning is executed for trip start times commencing with the end of the current activity e'. The FILTER-OPENING-HOURS() function further filters modes according to the attractor l opening hours based on estimated per-mode trip durations D, end of the current activity e' and the duration d of the next activity. Finally, SAMPLE-MCM selects one of the remaining modes returning also the associated trip duration $d_t \in D$.

Although DDAS was designed with an intention to eliminate as many expert rules as possible, due to a limited size of our training set (see Section 5.3), we had to enforce the following rule-based constraints:

1. Only modes available to the person are allowed. For example, the `car` mode is allowed only for those agents who actually have a car.
2. The `car` and `bike` mode is disabled when the current attractor l' is farther than 500 m from the car's or bike's current location. This applies to situations such as having a car or a bike left home using other modes since.
3. Force `car` or `bike` mode when they were used for a previous trip and the next attractor l is more than 500 m away. In other words, an agent leaving home using `car` or `bike` is forced to keep using the same mode, with an exception of short trips.

5. Validation

In this section, we validate an implementation of DDAS at a particular area of interest using the Validation Framework for Activity-Based Models (VALFRAM) (Drchal et al., 2016). We compare the validity of DDAS to the validity of an alternative called AgentPolis Scheduler (APS) (Čertický et al., 2015), which is a DDAS predecessor loosely based on ALBATROS scheduler (Arentze and Timmermans, 2000).

Table 1
Overview of the VALFRAM steps.

Validation task	Statistic
Activities in Time (A1) distributions of start times and durations for each activity	KS
Activities in Space (A2) distribution of each activity type in 2D space	MAE
Structure of Activities: Activity Count (A3a) activity counts within activity schedules	χ^2
Structure of Activities: Activity Sequences (A3b) activity counts within activity schedules	χ^2
Trips in Time: Modes by Time of Day (B1a) distribution of selected modes by time of day	χ^2
Trips in Time: Trip Times per Mode (B1b) distribution of travel times by mode	KS
Trips in Space (B2) distance between generated and real-world OD matrix	MAE
Mode for Target Activity Type (B3) distribution of selected transport mode for each type of target activity	χ^2
Attractor Visits (C1) distribution of generated attractor visits	MAE

5.1. Modelled scenario

Implementations of both the DDAS and APS scheduler target South Moravian Region of Czech Republic, populated by approximately 1.2 million citizens. An overview of the modelled area is depicted in Fig. 5. The red rectangle shows an area which we used to train AAM. Both models produce the activity schedules for a 24 h period representing a typical workday.

5.2. Validation method

Validation of an activity-based model as a whole is a non-trivial task. Fortunately, there exists a Validation Framework for Activity-Based Models (VALFRAM) (Drchal et al., 2016) – a general methodology for statistical assessment of the performance of activity-based models, such as DDAS. This subsection contains a short overview of VALFRAM, as well as a few proposals for updates and extensions.

VALFRAM defines a set of steps, where each step compares a probabilistic distribution of a selected aspect of the generated schedules to ground truth represented by validation data. When comparing continuous distributions (e.g., activity durations), Kolmogorov-Smirnov (KS) two-sample statistic (Hollander et al., 2013), defined as a maximum deviation between the empirical cumulative distribution functions, is applied. For discrete distributions (e.g., activity type counts), a well-established Pearson's χ^2 test statistic (Sokal and Rohlf, 1994) or Mean Average Error (MAE) is used. In all cases, lower values indicate a better alignment between the generated and the ground truth distributions.

Table 1 contains an overview of the VALFRAM steps including information on the specific statistics used. The steps can be divided into three groups: (A) those focusing on generated *activities* (denoted A1-A3b), (B) *trips* (B1a-B3) and (C) *attractors* (C1) where the latter was not part of the original VALFRAM as proposed in Drchal et al. (2016) and will be described below. Individual VALFRAM steps are designed to validate various *temporal*, *spatial* and *structural* properties of the generated schedules.

Note that some of the VALFRAM steps can produce more than one value. As an example, *Activities in Time (A1)* returns a value of KS statistic for each possible activity type (`sleep`, `work`, `school`, `shop` and `leisure`) for both activity durations and start times. These can be invaluable for getting deep insight into the internals of the scheduler, however, in other cases, their number might be overwhelming. When concise validation is needed, we only consider the average values. To get an average over KS or MAE statistics, one can simply use arithmetic mean. Unfortunately, it is not suitable for the χ^2. For χ^2, a correct approach would be to compute a single value of the statistic based on aggregated data. Using arithmetic mean may be misleading when categories are not balanced, as in our case: while each schedule contains two `sleep` activities, an average number of `school` activities per schedule is strictly less than one as only a fraction of the generated population belongs to students. For such unbalanced categories, we use the weighted arithmetic mean. We present both unweighted and weighted means in the experiments below.

Attractor Visits (C1) is a VALFRAM extension with regard to Drchal et al. (2016). It computes Mean Absolute Error (MAE) of normalized attractor visits $\boldsymbol{\nu}$ (as described in Section 4.5) comparing those generated to the ground truth. Another possibility might be to use Root Mean Square Error (RMSE) in place of the MAE. However, in practice, we found it to be too noisy due to higher sensitivity to outliers.

Note that unlike in original VALFRAM proposal (Drchal et al., 2016), here we use MAE for both spatial steps A2 and B2 as a replacement for χ^2 and RMSE. The reason for selecting MAE was the same as in the C1 case: the quadratic component present in the definitions of both measures was prone to outliers. Note that we normalize data prior to MAE computation in all cases. As an example: in the A2 (Activities in Space) the elements of both vector representing the region visits generated by the scheduler and the

corresponding vector based on the validation data sum to one.

Our validation data for A1-B3 was based on travel diaries containing 1850 schedules (6822 activities). Note that different dataset of the same structure was used to train ATM, ADM and MCM modules. The C1 step was performed using Foursquare attractor visit data (see Section 3.2) which involved 5781 `leisure` and 3106 `shop` attractors. AAM was trained on their subset of 855 and 495 attractors respectively.

5.3. Module training: ATM, ADM and MCM

To implement the DDAS, one has to train all four modules ATM, ADM, MCM and AAM. Here we provide information on training the first three modules. The last one, AAM, is discussed in the following section.

All the datasets used to train and evaluate ATM, ADM and MCM were extracted solely from the travel diary data described in Section 3.4. We used the following methodology to configure each of the three models: (1) The dataset was split into a training set and a testing set (80% and 20% of the data respectively).[14] (2) We performed a 10-fold cross-validation on the training set to select the best configuration. (3) The model was trained 50 times using the whole training set and evaluated on the test set. We searched for the best performing *maximum tree depth*, while all the other parameters related to tree training were set to default values of the Scikit-learn machine learning library (Pedregosa et al., 2011). We also searched for a subset of dataset features which maximized the model accuracy. The feature selection is a common practice in machine learning, that can reduce the dimensionality of the optimization problem (fitting trees to data in our case). The features in consideration were: `count`, `soc`, `reach` for ATM/ADM, and `count`, `soc`, `reach`, `mode counts` and `trip duration estimates` for MCM.

The classification datasets extracted from the travel diaries were highly imbalanced. For ATM, the class distribution was: `sleep` (48.4%), `none` (24.2%), `work` (10.5%), `shop` (6.8%), `leisure` (5.5%) and `school` (4.6%). For MCM, we ended up with: `car` (47.3%), `pt` (43.6%), `walk` (6.0%) and `bike` (3.1%). This led us to use F1 score instead of more common accuracy measure. Specifically, we used F1 score with micro setting (Sokolova and Lapalme, 2009) to evaluate the classification models ATM and MCM. Since ADM is a regression model, we employed the standard mean squared error (MSE).

Table 2 summarizes the results. It shows the number of samples per each model's dataset, the selected subset of the input features, the selected maximum tree depth as well as the averaged final model performance on the test data. Interestingly, the `trip duration estimates` were not advantageous for the MCM. We hypothesize that this result is caused by: (1) low spatial precision of our travel diaries which forced us to use the same averaging method to estimate the trip duration as is employed for the `reach` features, (2) path planner precision and (3) the aforementioned class imbalance and the limited size of the training set.

5.4. Module training: AAM

In this section, we discuss AAM training and model selection. Being based on the multi-objective approach, AAM is a significant departure from the standard supervised-learning approaches. The selection of a particular model becomes non-trivial. We used the following methodology:

- AAM models for both `leisure` and `shop` activity types were trained on data restricted to the relatively small subarea shown in Fig. 5 where it is indicated by the red rectangle. The subarea was chosen with an intent to be as representative of the whole modelled region as possible – it contains both densely and sparsely populated areas.
- The *demand* instances were defined by the *current* activity attractor and its distances to all possible attractors of the next activity d_{ij} as described in Section 4.5. As the size and spatial resolution of the travel diaries were very limited, we decided to extract this information from the schedules generated using the APS described in the Section 5.5. For the `shop` activity, the training set consisted of 495 attractors and 3235 demand instances. For `leisure`, these were 855 and 2661 respectively.
- The parameters of AAM models were optimized using Adam (Kingma and Ba, 2014), which is a well-established alternative to Stochastic Gradient Descend. Unlike SGD, Adam is robust with respect to its meta-parameters – the learning rates are adaptively adjusted based on past values of gradient in a per-parameter fashion. Adam was run with the recommended parameter settings: learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$ and zero decay.
- The f_θ was implemented as an MLP having two hidden layers of 32 ReLU neurons and an output layer of a single linear neuron.
- We executed three runs for each AAM setting in consideration. Each run consisted of 10000 epochs, while for each 100th epoch the model parameters were recorded and the model was evaluated by means of both losses (Eqs. (11) and (12)) computed for the whole modelled area.

Initially, we have empirically selected the value of the loss balancing parameter to $\alpha = 0.1$ for both `leisure` and `shop` AAMs. For this setting there was both 1) highest number of non-dominated solutions and 2) widest spread of solutions in the Pareto front, i.e., the setting generated solutions spanning from those preferring short distances (low $\mathcal{L}_D$) to those preferring precise visit distribution (low $\mathcal{L}_V$).

The second set of experiments aimed at the evaluation of the feature selection over the set of the *common features*, where we tested all eight combinations of `soc`, `reach` and `count` (similarly to Section 5.3). We have found that no configuration brought

[14] In case of classification tasks (ATM and MCM), we employed a stratified split to preserve the class balance.

Table 2
DDAS modules training and results.

Model	Dataset size	Selected features	Max. tree depth	Test performance
ATM	6730	`reach`, `soc`	6	0.87 (F1-score)
ADM	5101	`reach`, `soc`	9	4.15 (MSE)
MCM	1830	`count`, `reach`, `soc`, `mode counts`	8	0.83 (F1-score)

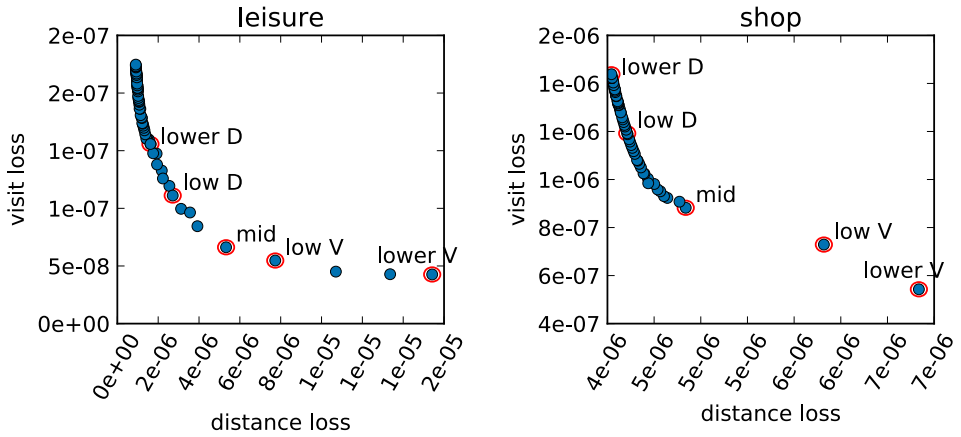


Fig. 6. Pareto fronts for (left) `leisure` and (right) `shop` AAM models. Red circles indicate the selected configurations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

significantly better results than the others and therefore we decided to omit the *common features* completely in the following analysis. Our hypothesis, which is a subject of future research is, that the problem is caused by using APS as a proxy to actual demand.

Fig. 6 shows the Pareto fronts for both `leisure` and `shop` AAM models trained using $\alpha = 0.1$. The red circles indicate five non-dominated configurations which were selected for further experiments, spanning from those having low values of $\mathcal{L}_D$ and higher values of $\mathcal{L}_V$ to the opposite.

Fig. 7 compares responses of the `leisure` AAM to the training data and to the whole South Moravian Region. The graphs show attractor visit probabilities as generated by AAMs (computed using Eq. (10)) for two best configurations (`low D` and `mid`) where the attractors are sorted by decreasing proportions of the actual visits, which are also shown. Note that `mid` configuration fits the actual data with higher precision as expected for a model with the lower value of $\mathcal{L}_V$. The `mid` model also shows better generalization when evaluated on the complete area. The results for the `shop` AAMs are presented in Fig. 8. One can see that the predictions are considerably noisier than in case of `leisure` although at least the highly visited attractors are still being discriminated from those less visited.

Additionally to this direct AAM evaluation, we executed several runs of the whole DDAS scheduler testing the preselected AAM configurations using VALFRAM. Based mostly on Trip in Time (B1a, B1b), Trip in Space (B2) and Attractor Visits (C1) steps we decided to use the `mid` configuration for the `leisure` AAM while the `low D` for the `shop` AAM for all following experiments.

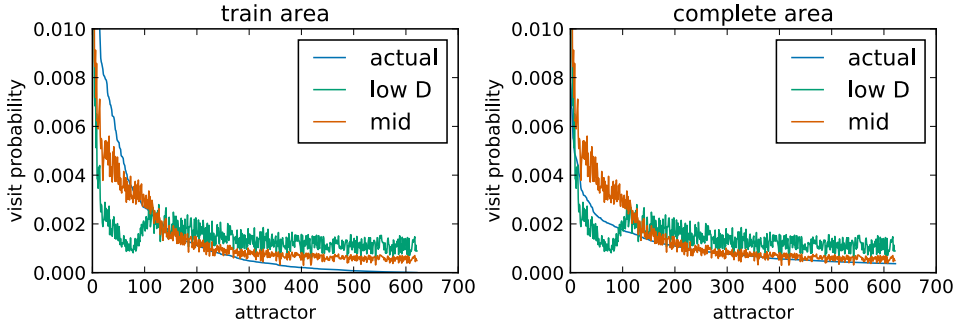


Fig. 7. Visit proportion of validation data and two settings of AAM for all 622 `leisure` train set attractors (left) and for the first 622 most visited attractors of the complete area (right). The attractors are sorted in a decreasing order of actual visits.

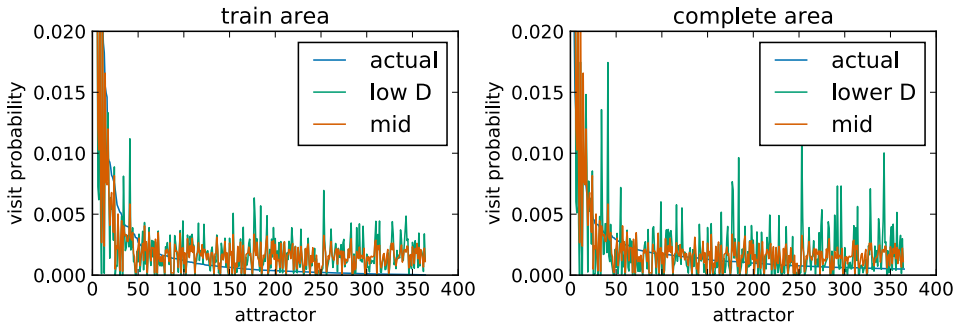


Fig. 8. Visit proportion of validation data and two settings of AAM for all 365 `shop` train set attractors (left) and for the first 365 most visited attractors of the complete area (right). The `shop` AAMs are noisier than their `leisure` counterparts (see Fig. 7).

5.5. AgentPolis scheduler as a baseline

We compare the validity of DDAS to an alternative scheduler called *AgentPolis Scheduler* (APS) (Čertický et al., 2015; Drchal et al., 2016), which is loosely based on ALBATROSS model (Arentze and Timmermans, 2000). Even though several of its modules use machine learning methods, they are connected by a large set of expert-defined rules and constraints. The scheduling in APS is a complex iterative process, in which the activities are repeatedly checked for being reachable by transport. To estimate the trip durations, as well as the detailed composition of the trips, we use the same multimodal path planner and timetable data as in our DDAS implementation. For more details please see (Čertický et al., 2015; Drchal et al., 2016).

5.6. Validation results

Comparing the validity of *Activities in Time (A1)* for activity durations, we get mean *KS* statistics of $MKS_{\text{DDAS}} = 0.069$ for DDAS and $MKS_{\text{APS}} = 0.24$ for APS. The situation is similar for means weighted over activity type count ($MKS^{w}_{\text{DDAS}} = 0.041$ and $MKS^{w}_{\text{APS}} = 0.16$).

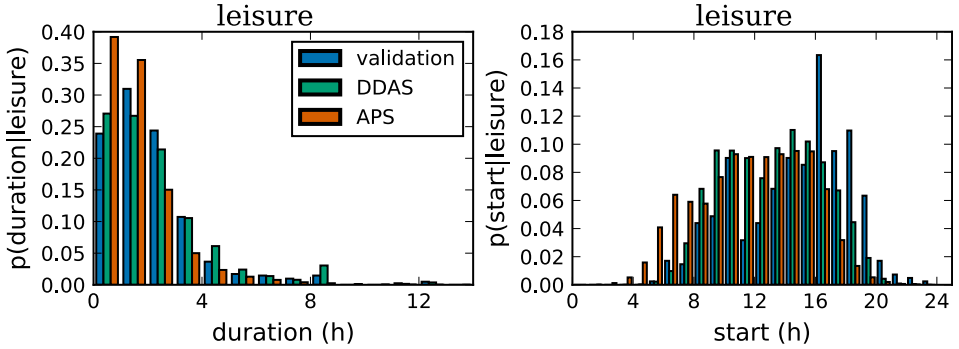
For the activity start times, DDAS also outperforms APS, although the difference is not that significant ($MKS_{\text{DDAS}} = 0.14$ for DDAS, $MKS_{\text{APS}} = 0.25$ for APS with the weighted versions $MKS^{w}_{\text{DDAS}} = 0.068$ and $MKS^{w}_{\text{APS}} = 0.13$).

A comparison of the distributions generated by both schedulers to the actual validation distribution for the `leisure` activity type is depicted in Fig. 9. One can see that DDAS fits both distributions better, although both schedulers struggle with fitting the late afternoon peaks of `leisure` activity starts.

The results for *Activities in Space (A2)* are summarized in Table 3. Performance of DDAS and APS is very similar. APS gives a slightly better results for `school` and `leisure` activity types. It is outperformed for `sleep`, `work` and `shop`. For the `sleep` activities, the difference is negligible.

VALFRAM result for the *Structure of Activities: Activity Count (A3a)* clearly favours DDAS ($\chi^2_{\text{DDAS}} = 538$) over APS ($\chi^2_{\text{APS}} = 7144$) for the `leisure` activity type. For the `shop` activities, APS gives a slightly better fit ($\chi^2_{\text{APS}} = 7144$, $\chi^2_{\text{DDAS}} = 7374$). The histograms for both activity types are depicted in Fig. 10.

The analysis of activity sequences realized by *Structure of Activities: Activity Sequences (A3b)* again favours DDAS ($\chi^2_{\text{DDAS}} = 1.58 \times 10^6$) over APS ($\chi^2_{\text{APS}} = 7.17 \times 10^6$). This is expected: DDAS' ATM directly fits the actual sequences of activity types while in APS the order of activities is established using expert-designed optimization process, where activities are iteratively being assigned to still empty intervals of the schedule.



(a) duration (b) start time

Fig. 9. VALFRAM: Activities in Time (A1) for the `leisure` activity. DDAS outperforms APS for both activity durations and start times. The difference is less pronounced for the latter. The "validation" values correspond to validation data (ground truth).

Table 3
VALFRAM: activities in Space (A2). Mean Average Error (MAE) is presented. Bold typeface values indicate better results.

	sleep	work	school	leisure	shop
MAE_{DDAS}	**4. 42 × 10^{-3}**	**3. 02 × 10^{-3}**	7.99 × 10^{-3}	1.23 × 10^{-2}	**6. 2 × 10^{-3}**
MAE_{APS}	4.45 × 10^{-3}	3.45 × 10^{-3}	**7. 07 × 10^{-3}**	**6. 72 × 10^{-3}**	7.35 × 10^{-3}

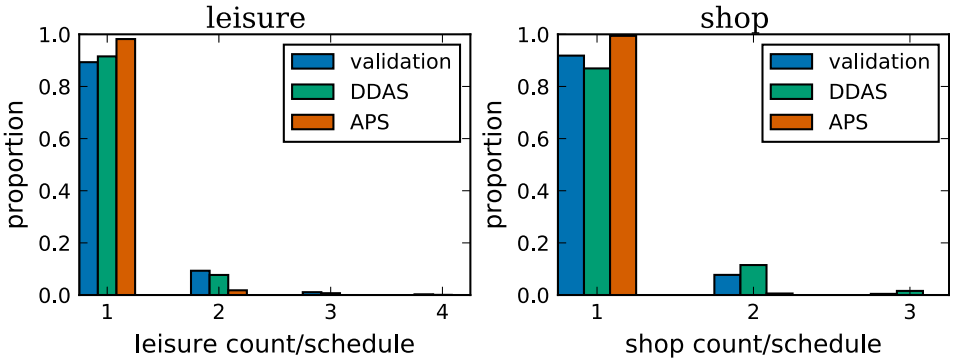


Fig. 10. VALFRAM: Structure of Activities: Activity Count (A3a). Average proportions of number of activity instances per schedule are shown for `leisure` and `shop`. The "validation" values correspond to validation data (ground truth).

Table 4
VALFRAM: Trips in Time: Modes by Time of Day (B1a). Values of χ^2 for all four-hour intervals are shown. Bold typeface values indicate better results.

	0:00–4:00	4:00–8:00	8:00–12:00	12:00–16:00	16:00–20:00	20:00–24:00
χ^2_{DDAS}	**91. 51**	**57420**	86070	**43300**	79520	13400
χ^2_{APS}	2753	60310	**6485**	84620	**8927**	**1075**

The validation of generated trips commences with *Trips in Time: Modes by Time of Day (B1a)* VALFRAM step, results of which are presented in Table 4 showing χ^2 values for all four hour intervals. Although both DDAS and APS achieves lower χ^2 for three intervals, the differences in favour of APS are more pronounced in most cases.

Trips in Time: Trip Times per Mode (B1b) VALFRAM step reported the lower (better) values of the mean *KS* statistic for APS than for DDAS ($MKS_{\text{APS}} = 0.24$ compared to $MKS_{\text{DDAS}} = 0.34$) which applied also for the weighted means ($MKS^w_{\text{APS}} = 0.22$ and $MKS^w_{\text{DDAS}} = 0.31$). When inspecting the individual modes we have found that DDAS outperforms APS only for `bike`. Fig. 11 shows the histograms of trip time distributions for `walk` and `bike` modes. Note that unlike APS, DDAS generates trips with higher durations for `walk`. Although we have found that this behaviour can be fixed by constraining the maximum duration of `walk` trips accordingly, which improved the results significantly (not shown here), we suggest that the problem should be rather addressed by obtaining more training data for MCM.

Results for *Trips in Space (B2)* which compares O-D matrices are very close for both schedulers ($MAE_{\text{DDAS}} = 3.83 \times 10^{-5}$ and

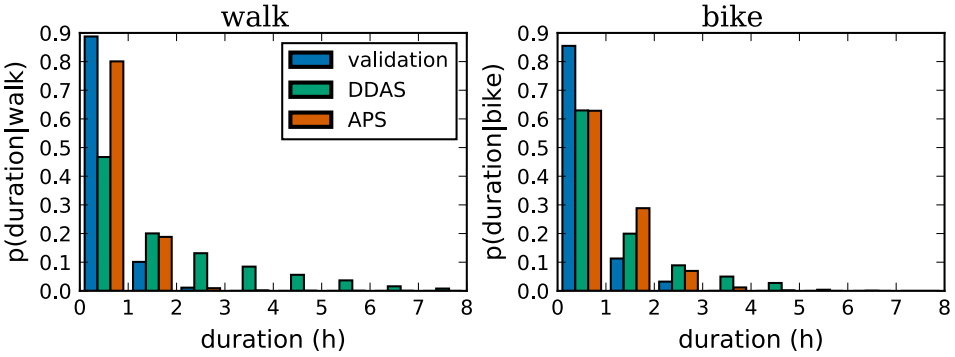


Fig. 11. VALFRAM: Trips in Time: Trip Times per Mode (B1b). Trip duration distributions are compared for `walk` mode (left) where DDAS gave the worst results and for `bike` mode (right) where it outperformed APS. The "validation" values correspond to validation data (ground truth).

Table 5
VALFRAM: mode for target activity type. Bold typeface values indicate better results.

	sleep	work	school	leisure	shop
χ^2_{DDAS}	145200	**19240**	**30440**	22740	23940
χ^2_{APS}	**43540**	39150	38050	**22720**	**12600**

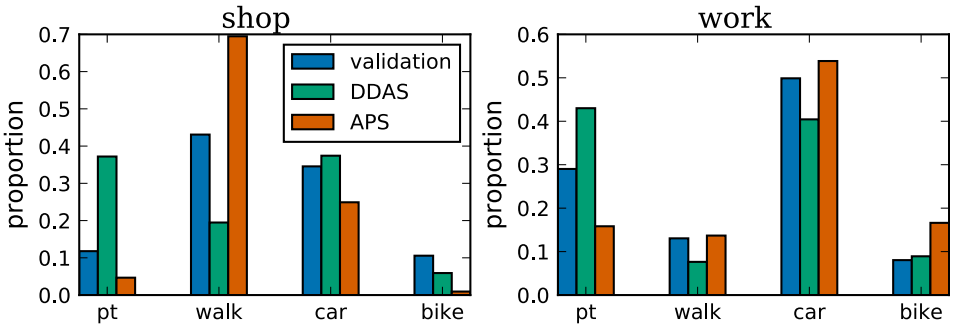


Fig. 12. VALFRAM: Mode for Target Activity Type (B3). Modal split for `shop` (left) and `work` (right) trip target activity types is shown. The "validation" values correspond to validation data (ground truth).

$MAE_{\text{APS}} = 3.79 \times 10^{-5}$).

The results of trip mode analysis realized by *Mode for Target Activity Type (B3)* are summarized in Table 5 showing χ^2 values. DDAS achieves a better fit for `work` and `school` activity types while worse for other (for `leisure` the difference is negligible). Fig. 12 compares mode distributions for validation and generated schedules for `shop` and `work` trip target activities. One can see that the modal split is not ideal for neither DDAS nor APS. We expect better results for both DDAS and APS when more data are available for MCM as discussed above (note that both schedulers' mode choice models are based on the same data).

The final VALFRAM step *Attractor Visits (C1)* compares the categorical distributions of actual attractor visit proportions to the generated ones. For `leisure` attractors we get $MAE_{\text{DDAS}} = 1.25 \times 10^{-4}$ and $MAE_{\text{APS}} = 2.64 \times 10^{-4}$ which was expected as APS attractor selection is based on relatively crude expert estimates only (Čertický et al., 2015; Drchal et al., 2016). Similarly for the `shop` attractors, DDAS achieves $MAE_{\text{DDAS}} = 4.02 \times 10^{-4}$ while APS error gets higher ($MAE_{\text{APS}} = 4.86 \times 10^{-4}$). In this latter case, the improvement of DDAS over APS is less significant, which can be explained by higher amount of noise observed while training the individual `shop` AAM (see Section 5.4).

To summarize, we performed overall validation of DDAS using VALFRAM, where we used DDAS' predecessor APS as the baseline. The results show that DDAS performs similarly to APS while its implementation is considerably simpler. DDAS brings significant improvement in attractor visit proportions (C1), while it is slightly inferior in cases where MCM has higher influence (B1a, B1b and B3). In these, however, neither DDAS nor APS perform particularly well. We expect that MCM would give much better results for larger and better-balanced training data.

6. Conclusion

Data-Driven Activity Scheduler (DDAS), presented in this paper, is a significant step towards fully data-driven activity-based models. The validity of our proof-of-concept implementation seems to be on par with alternative schedulers, despite being considerably less complex. Thanks to the simplicity of DDAS, we were able to provide a detailed description, including a pseudo-code, which should enable and encourage reproducibility and help establish a common baseline for future research of activity schedulers.

DDAS is composed of four core modules, one of which poses an important contribution by itself: The Activity Attractor Model (AAM) is used to select specific attractor coordinates for agent's activities, while the alternatives could only consider wider encompassing areas. Moreover, AAM is not limited to the area it was trained on. We have shown that it can be easily trained on a subarea containing only a small subset of attractors and then scaled to the whole area of interest.

Even though the expert-designed rules were almost eliminated in DDAS, some domain-specific knowledge is still involved in the structure of the model (e.g., AAM coordination with the external path planner). Design choices leading to this architecture were mostly influenced by: (1) unavailability of detailed data, such as trip diaries with nonaggregated spatial information, (2) small dataset sizes, and (3) a need to cooperate with external software components such as path planners.

Acknowledgement

This work was supported by the Technology Agency of the Czech Republic, program "Competence Centres" (Grant No. TE01010155).

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center - LM2015070".

The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

This publication was further supported by the European social fund within the framework of realizing the project "Support of inter-sectoral mobility and quality enhancement of research teams at Czech Technical University in Prague", CZ.1.07/2.3.00/30.0034, period of the project's realization 1. 12. 2012-30. 6. 2015 and by the European Union Seventh Framework Programme FP7/2007–2013 (grant agreement No. 289067).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.trc.2018.12.002.

References

Adu-Gyamfi, Y.O., Asare, S.K., Sharma, A., Titus, T., 2017. Automated vehicle recognition with deep convolutional neural networks. Transp. Res. Rec. 2645 (1), 113–122.

Alsger, A., Tavassoli, A., Mesbah, M., Ferreira, L., Hickman, M., 2018. Public transport trip purpose inference using smart card fare data. Transport. Res. Part C: Emerg. Technol. 87, 123–137.

Arentze, T., Timmermans, H., 2000. Albatross: A Learning Based Transportation Oriented Simulation System. Eirass Eindhoven.

Auld, J., Mohammadian, A.K., 2012. Activity planning processes in the agent-based dynamic activity planning and travel scheduling (adapts) model. Transport. Res. Part A: Policy Pract. 46 (8), 1386–1403.

Aziz, H.M.A., Park, B.H., Morton, A., Stewart, R.N., Hilliard, M., Maness, M., 2018. A high resolution agent-based model to support walk-bicycle infrastructure investment decisions: a case study with New York City. Transport. Res. Part C: Emerg. Technol. 86, 280–299.

Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., 2011. Sumo – simulation of urban mobility. In: The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain.

Ben-Akivai, M., Bowman, J.L., Gopinath, D., 1996. Travel demand model system for the information era. Transportation 23 (3), 241–266.

Bhat, C., Guo, J., Srinivasan, S., Sivakumar, A., 2004. Comprehensive econometric microsimulator for daily activity-travel patterns. Transport. Res. Rec.: J. Transport. Res. Board (1894), 57–66.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Bou Mjahed, L., Mittal, A., Elfar, A., Mahmassani, H.S., Chen, Y., 2017. Exploring the role of social media platforms in informing trip planning: case of Yelp.com. Transp. Res. Rec. 2666 (1), 1–9.

Bowman, J.L., Bradley, M.A., Gibb, J., 2006. The sacramento activity-based travel demand model: estimation and validation results. In: European Transport Conference.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32. https://doi.org/10.1023/A:1010933404324.

Castiglione, J., Bradley, M., Gliebe, J., 2015. Activity-based travel demand models: A primer. No. SHRP 2 Report S2-C46-RR-1.

Čertický, M., Drchal, J., Cuchý, M., Jakob, M., 2015. Fully agent-based simulation model of multimodal mobility in european cities. In: 2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, pp. 229–236.

Chaniotakis, E., Antoniou, C., Aifadopoulou, G., Dimitriou, L., 2017. Inferring activities from social media data. Transp. Res. Rec. 2666 (1), 29–37.

de Dios Ortuzar, J., Willumsen, L.G., 2011. Modelling Transport, fourth ed. John Wiley & Sons.

Dianat, L., Habib, K.N., Miller, E.J., 2017. Two-level, dynamic, week-long work episode scheduling model. Transp. Res. Rec. 2664 (1), 59–68.

Doherty, S.T., 2000. An activity scheduling process approach to understanding travel behavior. In: 79th Annual Meeting of the Transportation Research Board, Washington, DC.

Drchal, J., Čertický, M., Jakob, M., 2016. VALFRAM: validation framework for activity-based models. J. Artif. Societ. Soc. Simul.(19(3)). http://jasss.soc.surrey.ac.uk/19/3/5.html .

Gadzinski, J., 2018. Perspectives of the use of smartphones in travel behaviour studies: findings from a literature review and a pilot study. Transport. Res. Part C: Emerg. Technol. 88, 74–86.

Gärling, T., Kwan, M.-p., Golledge, R.G., 1994. Computational-process modelling of household activity scheduling. Transport. Res. Part B: Methodol. 28 (5), 355–364.

Geoffrion, A.M., 1976. The purpose of mathematical programming is insight, not numbers. Interfaces 7 (1), 81–92.

Ghasri, M., Rashidi, T.H., Waller, S.T., 2017. Developing a disaggregate travel demand system of models using data mining techniques. Transport. Res. Part A: Policy Pract. 105, 138–153.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org .

Hafezi, M.H., Liu, L., Millward, H., 2018. Learning daily activity sequences of population groups using random forest theory. Transp. Res. Rec 0361198118773197.

Hall, R.W., 1986. Discrete models/continuous models. Omega 14 (3), 213–220.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385. <http://arxiv.org/abs/1512.03385>.

Hilgert, T., Heilig, M., Kagerbauer, M., Vortisch, P., 2017. Modeling week activity schedules for travel demand models. Transp. Res. Rec. 2666 (1), 69–77.

Hollander, M., Wolfe, D.A., Chicken, E., 2013. Nonparametric Statistical Methods, third ed. Wiley.

Janssens, D., Wets, G., Brijs, T., Vanhoof, K., Arentze, T., Timmermans, H., 2004. Improving performance of multiagent rule-based model for activity pattern decisions with bayesian networks. Transport. Res. Rec.: J. Transport. Res. Board 1894, 75–83.

Jones, P.M., Dix, M.C., Clarke, M.I., Heggie, I.G., 1983. Understanding Travel Behaviour. Gower Publishing, Brookfield, VT United States.

Kashiyama, T., Pang, Y., Sekimoto, Y., 2017. Open PFLOW: creation and evaluation of an open dataset for typical people mass movement in urban areas. Transport. Res. Part C: Emerg. Technol. 85, 249–267.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6. <http://arxiv.org/abs/1412.6980>.

Kitamura, R., 1983. Sequential, history-dependent approach to trip-chaining behavior. Transp. Res. Rec. 944, 13–22.

Kitamura, R., Chen, C., Pendyala, R., 1997. Generation of synthetic daily activity-travel patterns. Transport. Res. Rec.: J. Transport. Res. Board (1607), 154–162.

Klügl, F., 2009. Agent-based simulation engineering. Ph.D. thesis, Habilitation Thesis. University of Würzburg.

Kotsiantis, S., Kanellopoulos, D., 2006. Discretization techniques: a recent survey. GESTS Int. Trans. Comput. Sci. Eng. 32 (1), 47–58.

Kuflik, T., Minkov, E., Nocera, S., Grant-Muller, S., Gal-Tzur, A., Shoor, I., 2017. Automating a framework to extract and analyse transport related social media content: the potential and the challenges. Transport. Res. Part C: Emerg. Technol. 77, 275–291.

Langevin, A., Mbaraga, P., Campbell, J.F., 1996. Continuous approximation models in freight distribution: an overview. Transport. Res. Part B: Methodol. 30 (3), 163–188.

Liu, L., Chen, R.-C., 2017. A novel passenger flow prediction model using deep learning methods. Transport. Res. Part C: Emerg. Technol. 84, 74–91.
McNally, M.G., 1986. On the formation of household travel/activity patterns: a simulation approach. Tech. rep.
McNally, M.G., 2008. The four step model. Center for Activity Systems Analysis.
Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, third ed. Springer-Verlag, London, UK. http://dl.acm.org/citation.cfm?id=229930 .
Miller, E., Roorda, M., 2003. Prototype model of household activity-travel scheduling. Transport. Res. Rec.: J. Transport. Res. Board (1831), 114–121.
Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., Bowling, M., 2017. DeepStack: expert-level artificial intelligence in heads-up no-limit poker. Science 356 (6337), 508–513. https://doi.org/10.1126/science.aam6960.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Rashidi, T.H., Abbasi, A., Maghrebi, M., Hasan, S., Waller, T.S., 2017. Exploring the capacity of social media data for modelling travel behaviour: opportunities and challenges. Transport. Res. Part C: Emerg. Technol. 75, 197–211.
Rasouli, S., Timmermans, H., 2014a. Activity-based models of travel demand: promises, progress and prospects. Int. J. Urban Sci. 18 (1), 31–60.
Rasouli, S., Timmermans, H.J., 2014b. Using ensembles of decision trees to predict transport mode choice decisions: effects on predictive success and uncertainty estimates. Eur. J. Transp. Infrastruct. Res. 14 (4), 412–424.
Samek, W., Wiegand, T., Müller, K.-R., Aug 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv:1708.08296 [cs, stat]ArXiv: 1708.08296. <http://arxiv.org/abs/1708.08296>.
Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., 2017. Mastering the game of go without human knowledge. Nature 550 (7676), 354.
Smith, L., Beckman, R., Anson, D., Nagel, K., Williams, M.E., 1995. Transims: Transportation analysis and simulation system. In: 5th National Conference on Transportation Planning Methods Applications-Volume II.
Sokal, R.R., Rohlf, F.J., 1994. Biometry: The Principles and Practices of Statistics in Biological Research, third ed. W.H. Freeman.
Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inform. Process. Manage. 45 (4), 427–437.
Turner, R., 2016. A model explanation system. In: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6.
Wilson, N.H.M., Sussman, J., Goodman, L., Hignnet, B., 1969. Simulation of a computer aided routing system (cars). In: Proceedings of the third conference on applications of simulation. Winter Simulation Conference, pp. 171–183.
Wu, X., Guo, J., Xian, K., Zhou, X., 2018. Hierarchical travel demand estimation using multiple data sources: a forward and backward propagation algorithmic framework on a layered computational graph. Transport. Res. Part C: Emerg. Technol. 96, 321–346.
Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., Zweig, G., 2016. Achieving human parity in conversational speech recognition. CoRR abs/1610.05256. <http://arxiv.org/abs/1610.05256>.