J Geograph Syst (2001) 3:325–346

Journal of Geographical Systems

Deriving rules from activity diary data: A learning algorithm and results of computer experiments

Theo A. Arentze[1], Frank Hofman[2], Harry J.P. Timmermans[1]

[1] Urban Planning Group EIRASS, Eindhoven University of Technology, P.O.Box 513, 5600 MB, Eindhoven, The Netherlands (e-mail: eirass@bwk.tue.nl)
[2] Ministry of Transport, Public Works and Water Management, P.O.Box 1031, 3000 BA, Rotterdam, The Netherlands

Received: 31 January 2001 / Accepted: 13 September 2001

Abstract. Activity-based models consider travel as a derived demand from the activities households need to conduct in space and time. Over the last 15 years, computational or rule-based models of activity scheduling have gained increasing interest in time-geography and transportation research. This paper argues that a lack of techniques for deriving rules from empirical data hinders the further development of rule-based systems in this area. To overcome this problem, this paper develops and tests an algorithm for inductively deriving rules from activity-diary data. The decision table formalism is used to exhaustively represent the theoretically possible decision rules that individuals may use in sequencing a given set of activities. Actual activity patterns of individuals are supplied to the system as examples. In an incremental learning process, the system progressively improves on the selection of rules used for reproducing the examples. Computer experiments based on simulated data are performed to fine-tune rule selection and rule value update functions. The results suggest that the system is effective and fairly robust for parameter settings. It is concluded, therefore, that the proposed approach opens up possibilities to derive empirically tested rule-based models of activity scheduling. Follow-up research will be concerned with testing the system on empirical data.

Key words: Activity-based models, activity scheduling, rule-based systems, decision tables, inductive machine learning, adaptive systems

1 Introduction

Activity-based models of trip generation and distribution have recently regained considerable popularity. These models consider travel as derived demand. In this view, travel is not an end in itself but a consequence of the participation of households in activities which take place at spatially dispersed locations (Cullen and Godson 1975). Hence, the aim of activity-based travel demand models is to describe and predict activity patterns of

households as a function of the availability of facilities in space-time environments which are necessary for performing certain activities. Many different approaches have been suggested. An overview is given in Ettema and Timmermans (1997), Kwan and Golledge (1997) and Bhat and Koppelman (1999).

One of the earliest approaches is based on theoretical notions of space-time geography developed by Hägerstrand (1970). Proposed models in this tradition incorporate space-time constraints which define feasible activity patterns given a list of programmed activities, available locations, opening times of facilities and structure of the transportation system. Typically, a combinatorial algorithm is used to generate feasible activity patterns. The number and sometimes also the quality of feasible patterns are then used as measures of the flexibility of the space-time environment. Proposed measures have been applied to evaluate impacts of policies affecting the time-space environment on activity patterns. One of the first models in this tradition is Lenntorp's PESASP model (Lenntorp 1976). A more recent example is the CARLA model (Jones et al. 1983).

As these models do not attempt to model activity choice behaviour, utility-based models have been proposed as an alternative approach. Most of these models involve an extension of conventional (nested) logit models that have been applied in transportation research since the mid seventies to predict destination and mode choice behaviour based on trip or tour data. Thus, the mathematical framework for these models has remained the same, but data on trips and/or tours have been replaced by data on activity patterns. As recent examples, Ben-Akiva and Bowman (1995) and Ettema et al. (1997) proposed a nested-logit model structure in which the choice of trips is embedded within the choice of complete activity patterns. Recker et al. (1986a, b) earlier proposed a utility-based model called STARCHILD which incorporates more complex mechanisms for defining and reducing choice-sets for partial choices.

Besides discrete choice models, discrete-continuous models of activity generation and scheduling have been proposed within a utility-maximising framework. These models treat time engaged in activities as a continuous dependent variable while taking into account the discrete nature of choices to engage in activities. Kitamura (1984) and Kitamura et al. (1996b) developed a discrete-continuous model considering the engagement in, and time allocation to, two types of discretionary activities and formulated the model as a doubly-censored Tobit model.

Both the time-geographical and utility-based approaches have been criticised, however, for their inability to account for cognitive constraints in human decision making. Implicitly, the combinatorial algorithms assume that individuals have complete knowledge of their environment and consider all feasible activity patterns. On the other hand, utility-based models allow bounded rationality in terms of a reduced choice set for each choice involved. Still in the stage of making a choice, these models assume optimal behaviour in assuming that individuals invariably compare choice alternatives on a set of attributes and choose the one that maximises utility.

Computational process models have been introduced as an alternative approach, as reviewed in Gärling et al. (1994), Golledge et al. (1994) and Kwan and Golledge (1997). These models attempt to model the underlying

individual problem solving process. They assume that the process is controlled by heuristic decision rules, which have evolved and are continuously adapted through learning processes. Thus, a heuristic rule does not necessarily produce optimal outcomes, but rather actions that have yielded satisfactory outcomes under similar conditions in the past. Pendyala et al. (1997) proposed a dynamic micro-simulator of household activities and travel called AMOS. The system simulates the scheduling and adaptation of schedules and resulting travel behaviour of individuals and households. Kitamura and Fujii (1996b) proposed the PCATS model. This model adopts Hägelstrand's time-space prism notion and simulates activity choices sequentially while considering the prism constraints. Certain components of AMOS and PCATS are very similar to another model, proposed by Ettema et al. (1994), which also assumed a heuristic and sequential decision making process. SMASH, as their system is called, is designed to simulate the process of activity scheduling.

Although current computational process models intend to model cognitive processes, their theoretical foundation in cognitive psychology is weak. Production systems, first introduced by Newell and Simon (1972), have been widely recognised as a natural formalism to model human problem solving process. Production systems consist of a set of IF-THEN rules (called productions), a short-term memory and a control mechanism. The control mechanism matches items of short-term memory with conditions of rules in long term memory and executes the action of the rule whose condition is met. These notions were first applied in the area of activity scheduling by Hayes-Roth and Hayes-Roth (1979). Gärling et al. (1989) developed SCHEDULER which is a rule-based system using some of the earlier concepts. In a later study, Golledge et al. (1994) demonstrated how this system could be interfaced with a geographic information system. Vause (1997) argued that a rule-based system for activity scheduling should consist of specialised rules for controlling the problem solving process (meta-rules), restricting individual's choice sets and making decisions. Kwan (1997) developed GISICAS, a system for supporting individual's travel decisions based on a geographic information system for representing the spatial-environment and heuristic rules for activity scheduling and finding locations and routes for performing the activities.

Although rule-based systems have attracted much attention, their use as predictive models of activity scheduling has been limited, arguably, because a statistical technique for deriving the rules from activity data is lacking (Golledge et al. 1994). To date, rule-based models have typically been derived deductively using common sense. In other application areas, qualitative knowledge acquisition techniques developed in cognitive sciences (i.e., AI and cognitive psychology), such as for example interviewing and think aloud protocols, have been used. However, the qualitative techniques necessarily rely on the case-study approach and lack a mechanism to optimise models in terms of a quantitative goodness-of-fit measure. Albatross, an activity-scheduling model developed by the authors in previous work, is the only operational rule-based model derived from data (Arentze and Timmermans 2000, 2001). For each step in a pre-defined sequential decision process, a decision tree induction method is used to construct a decision tree that best fits activity diary data of a sample of individuals. Although this approach

overcomes the problems of deductive and qualitative methods, it relies on optimising decision models at the level of individual choice facets and, therefore, does not guarantee an optimal fit at the level of complete activity schedules. In short, as Golledge et al. (1994) argued, appropriate statistical techniques for estimating and calibrating computational process models of activity scheduling, are yet to be defined.

The purpose of the present study, therefore, is to develop and test a learning algorithm for inductively deriving rules from activity diary data. The learning algorithm proposed here uses the decision table formalism to exhaustively represent theoretically possible decision rules that individuals may use. Activity diary data of a representative sample of individuals are used to train the system. Through an incremental learning process the system progressively improves on the selection of rules for predicting the activity patterns. Hereby, a measure of distance between generated and observed activity sequences is used to provide feedback.

The paper is structured as follows. First, Sect. 2 discusses the proposed rule-based scheduling system. Then, Sect. 3 focuses on the learning algorithm proposed for deriving decision rules from activity diary data. The numerical properties of the algorithm are tested in a series of computer experiments using simulated data. The section that follows discusses the results of these experiments. Finally, the last section summarises the major conclusions and discusses possible directions for further research.

2 The modelling approach

Scheduling the activities of a given individual for a given day requires a number of interrelated decisions on several dimensions, such as the nature of the activities (what?), the timing of the activities (when?), the location where to perform the activities (where?), the transport mode used for travelling to out-of-home activities (with which mode?) and accompanying person(s) (with whom?). In the model that we propose, these decisions are made in a sequential process. Furthermore, the model assumes that individuals use decision heuristics in this step-wise process. In the present study, we consider as an illustrative case the scheduling problem in a more narrow sense. This problem involves determining the sequence in which the activities are going to be conducted. We assume that such scheduling decisions take place in a relatively early stage of the broader scheduling process, as a first step in determining the timing of activities.

2.1 Model components

The scheduling problem (in a narrow sense) can be described as finding for a given set of possible activities AP the ordered set $S \subseteq A$ which defines the selection and sequence of activities that are actually scheduled to take place. In the proposed system, the problem solving process is controlled by a built-in algorithm. The algorithm is initialised with an empty set S and builds a sequence by placing activities one-by-one from A into S, as follows:

(1) Determine the priority of each activity $a \in AP$ to add to S;
(2) for each $a \in AP$, from high to low priority;
(2.1) determine the set of feasible positions of a in the current set S;
(2.2) if there are no feasible positions, then evaluate the next activity;
(2.3) determine the priority of each feasible position of a in S;
(2.4) add a to S in the position with the highest priority;
(3) revise S by re-ordering activities.

The above algorithm is based on assumptions about how individuals solve scheduling (i.e., sequencing) problems. Constraints and preferences related to decision steps are represented by rules. With respect to the nature of the proposed rules various assumptions are made.

First, individuals may use different scheduling strategies resulting in different ways of ranking activities for insertion in the schedule (Step (1)). For example, individuals may first add the primary activity (e.g., work or school) for that day to the schedule and next determine the position of other activities relative to this activity (e.g., a social visit before or after work/school). Given the sequential character of the process, the chosen order in which activities are scheduled may affect the final outcome and therefore is an important decision element.

Second, assessing the feasibility of optional positions for a given activity (Step (2.1)) is based on knowledge about constraints imposed by the physical and institutional environment of the individual. Generally these include:

(1) *institutional constraints*, such as opening hours, influence the earliest and latest possible times to implement a particular activity.
(2) *household constraints*, such as bringing children to school, dictate when particular activities need to be performed and others cannot be performed.
(3) *spatial constraints* also have an impact in the sense that either particular activities cannot be performed at particular locations, or individuals have incomplete or incorrect information about the opportunities that particular locations may offer.
(4) *time constraints* limit the number of feasible activity patterns in the sense that activities do require some minimum duration and both the total amount of time and the amount of time for discretionary activities is limited.
(5) *spatial-temporal constraints* are critical in the sense that the specific interaction between an individual's activity program, the individual's cognitive space, the institutional context and the transportation environment may imply that an individual cannot be at a particular location at the right time to conduct a particular activity.

Supposedly, individuals evaluate relevant constraints of these types to determine the feasibility of each current schedule position for the activity under concern.

Third, individuals may have preferences for certain position-activity combinations (Step (2.3)).These may include (i) a preference for a specific time of the day to conduct the activity (e.g., shopping in morning hours), (ii) a preference for a specific sequence of conducting activities irrespective the time of the day (e.g., out-of-home activities before in-home activities), (iii) a

preference for specific linkages between activities (e.g., shopping after work on the route to home), etc.

The result of the above steps is a preliminary schedule. The *last step* (Step (3)) considers re-scheduling decisions to optimise the schedule. Also, in later stages of scheduling and even in the schedule implementation stage, the schedule may be revised, e.g., to adapt the schedule to accumulated information about available options or unforeseen events. Re-scheduling is based on the same types of preference and constraint knowledge that also underlies schedule construction.

Finally, the model assumes that the choice of heuristics for ranking of activities and schedule positions is dependent on the specific learning history of the individual. Decision rules are continuously evaluated and adapted dependent on the success of generated schedules in attaining the individual's objectives. The rule-based model intends to account for possible heterogeneity in preferences by considering the household and individual as potentially discriminating conditions for the success of rules. In other words, individual and household attributes are included as potentially relevant condition variables in the model.

2.2 Knowledge representation

Constraint rules and decision rules are treated distinctly in the proposed system. Constraint rules are a-priori specified by the system designer and form a fixed component of the system. This reflects the assumption that constraint rules represent basic knowledge about environments which is relatively stable over time and invariant across individuals. Space-time constraints are particularly relevant for activity-travel choices. In the present model, any set of constraints can be incorporated. Since the choice of constraint rules does not affect the learning algorithm, we will not further elaborate this part of the system here.

Decision rules, on the other hand, are represented in the form of decision tables which are optimised on observed behaviour. This reflects the assumption that decision rules represent the kind of problem solving strategies and preferences that tend to vary across individuals and are subject to adaptation over time.

The decision table (DT) is a well-established technique for structuring decision problems, but has only recently also been used as a formalism for representing knowledge in rule-based systems (e.g., Arentze et al. 1996; Lucardie 1994; Vanthienen and Wets 1994). A DT assumes a set of potentially relevant continuous or discrete condition variables, C_i for $i = 1 \ldots n$, and a discrete action variable, A, as given. The discrete values of the action variable represents the alternative actions. Given the domains of condition variables and action variables, the DT defines an exhaustive set of mutually exclusive decision rules. The general format of a decision rule can be described as:

$$\text{if } CS_{1j} \wedge CS_{2j}, \ldots CS_{nj} \text{ then do } A_j$$

where CS_{ij} represents the state of the i-th condition variable in the j-th rule and A_j the action alternative prescribed by the j-th rule. A condition state is defined as a subset of the given domain CD_i of the condition variable. Not necessarily every condition variable is relevant in every rule. When a

condition variable is irrelevant, the condition state is set to the condition domain, $CS_{ij} = CD_i$, implying that every possible condition value satisfies the condition in that case. The exhaustiveness and exclusiveness properties impose constraints on the definition of condition states. These properties make sure that the model is complete and consistent in the sense that each combination of possible condition values matches with exactly one rule of the set (for a formal definition of these properties see Wets 1998).

Figures 1 and 2 illustrate the application of the DT in the proposed scheduling system. The two DTs represent rules for assigning priorities to activities for adding to the schedule (Fig. 1) and rules for evaluating alternative schedule positions for each activity (Fig. 2). The two sections of the DT represent the conditions (upper part) and the actions (lower part) of the rules. In the present application, the action section represents alternative scheduling strategies that individuals in the population under investigation may use. The condition section represents the context variables that may affect the choice of a strategy. Relevant context variables may include characteristics of the individual, the activity program, the day of the week, etc. In the example, the first condition variable represents a household type with possible values type I and type II according to some classification. The second variable represents time pressure on the schedule. The columns represent conditional strategies. For example, if the time pressure on a schedule is high individuals may display a preference for clustering out-of-home activities, with the aim to create the conditions for trip chaining. In contrast, if the time

DT1

C1	Household type	I		II	
C2	Time pressure on schedule	Low	High	Low	High
A1	In-home activities first	X	–	–	–
A2	Out-home activities first	–	X	–	–
A3	Mandatory activities first	–	–	X	–
A4	Non-mandatory activities first	–	–	–	X
A5	From low to high flextime	–	–	–	–
A6	From high to low flextime	–	–	–	–

Fig. 1. Arbitrary decision table used for *activity* ranking in the simulations. X marks the rules used for generating data

DT2

C1	Household type	I		II	
C2	Time pressure on schedule	Low	High	Low	High
A1	In-home activities first	–	–	–	X
A2	Out-home activities first	–	–	X	–
A3	Mandatory activities first	–	X	–	–
A4	Non-mandatory activities first	X	–	–	–
A5	Link out-home activities	–	–	–	–
A6	Link in-home activities	–	–	–	–

Fig. 2. Arbitrary decision table used for activity-*position* ranking in the simulations. X marks the rules used for generating data

pressure is low the availability of ample time for completing a given activity may be the dominant criterion. At the same time, the strategies used may differ, for example, between households with and without children. Thus, each column in a DT represents a conditional scheduling strategy.

The advantage of the DT over unstructured rule sets is that DTs support a systematic verification of the consistency and completeness of the model. Consistency is verified by making sure that the condition states (columns) are mutually exclusive. Completeness is checked by ensuring that condition states cover the whole domain of a condition variable. Another motivation for using the technique in the present context is that DTs facilitate the development of algorithmic methods of inductive learning, as will be argued in the next section.

3 The learning algorithm

3.1 The problem

Our approach is based on the assumption that it is possible to identify for each scheduling decision the exhaustive set of possible strategies (actions) that individuals may use. Furthermore, we assume that it is possible to specify a-priori for each decision the potentially relevant condition states (columns). Informally, the search problem can be defined as identifying the action for each DT and each column within DTs that maximises a given measure of goodness-of-fit on a given set of observations.

To define this problem more formally, let $C_{i(t),h}$ be the observed value of the $i(t)$-th condition variable of the t-th DT for the h-th individual, $\mathrm{A}_{j(t),k(t)}$ a binary variable indicating whether or not the $k(t)$-th action alternative in the $j(t)$-th column of the t-th DT is to be selected, S_h the observed schedule of the h-th individual and RBS the rule-based system consisting of DTs t. RBS generates a schedule for each individual and, thus, defines the relationship:

$$RBS(C_{i(t),h}, A_{j(k),k(t)}) \Rightarrow S'_h \tag{1}$$

where S'_h is the generated schedule.

Assuming further a measure of similarity, $f(S', S)$, between any two schedules S' and S, the optimisation problem can be defined as:

$$\begin{aligned} \text{maximise}_{\{A_{j(t),k(t)}\}} \quad & \sum_h f\left(S'_h, S_h\right) \\ \text{while} \quad & \sum_{k(t)} A_{j(t),k(t)} = 1 \quad \forall t, j(t) \end{aligned} \tag{2}$$

The constraint condition states that in every column at least one and not more than one action is selected. In words, the problem is to find for each DT and for each DT column the action among a pre-specified set of alternatives that maximises aggregate similarity between generated and observed schedules.

3.2 Specification of the algorithm

The problem can be conceptualised as one of training a rule-based system based on examples (i.e., observed schedules). This so-called supervised learning has received much attention in machine learning and knowledge discovery

literature. With regard to rule-based systems, this work has focused on a particular subclass of problems, namely the classification of instances given a set of discrete classes (for a review of the field, see Fayyad 1996). Proposed systems induce a list of decision rules or a decision tree from supplied examples. Decision tree induction systems are among the most widely studied and applied systems. These systems construct a decision tree by recursively splitting a sample on attributes until all cases within groups are instances of a single class or no further improvements are possible. The splitting criterion used differs between systems. For example, C4.5 (Quinlan 1993) and CART (Breiman et al. 1984), which are the most popular decision tree induction systems, use an information gain and Gini measure as a splitting criterion respectively.

The multi-facet decision problem we are dealing with here cannot be reduced to a series of single facet classification problems, because there is no information about what the 'true' action at the individual decision step level is. The only information available is a quantitative measure of the similarity between an observed and generated schedule that is the result of multiple decision tables applied in a sequence. Even though the present learning problem is of the supervised type, the latter property implies that the model has characteristics in common with unsupervised, re-inforcement learning systems. Reinforcement learning is learning based on feedback on actions, in terms of a varying amount of reward, while the system is interacting with the environment. Reinforcement learning is a relatively recent area of AI. For a review of the field see Sutton and Barto (1998). Reinforcement learning systems that are of interest here incorporate a rule-value function, a rule-selection function and a feedback function. The first function incrementally updates the value of a rule each time feedback is received and the second function determines the selection of a rule in each new case based on the current value distribution. The learning algorithm proposed in this section uses this framework and specifies an appropriate incremental rule-value update function, a rule-selection function and feedback function for the present problem.

To develop the system, we conceptualise action alternatives $k(t)$ within columns $j(t)$ of DTs t as competing rules, namely rules having the same condition, but different actions. An error parameter is attached to each rule. This parameter indicates the past success of the rule in reproducing observed cases. Each time a case is to be processed, the system selects a rule from each competing rule set, as a probabilistic function of error values. Each time after processing a case, the system updates error values of the selected rules based on feedback in terms of the similarity measure. More formally, the algorithm can be described as:

(1) supply the system with the first case h;
(2) select an action alternative in each column $j(t)$ of each DT t based on a probability function of error values;
(3) generate a schedule given the $C_{i(t),h}$-data of the case;
(4) calculate the similarity measure with the observed schedule, $e_h = f(S'_h, S_h)$;
(5) update the error value of each selected action alternative, given e_h;
(6) repeat the procedure for the next case.

The data flows are schematically shown in Fig. 3. This algorithm describes an incremental learning process in which feedback information is accumu-

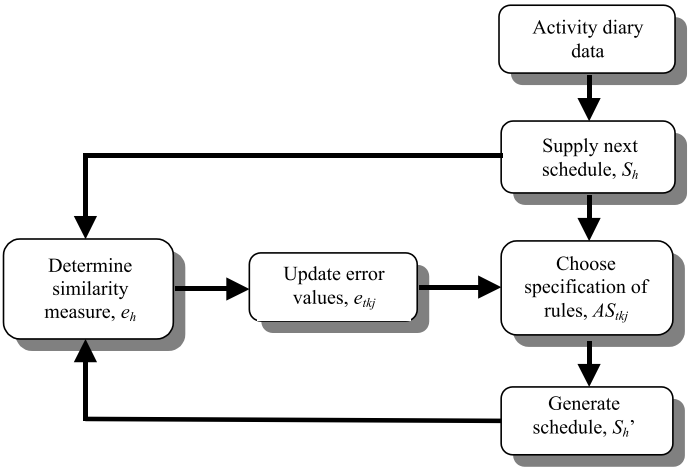

Fig. 3. Data flows in a schedule generation-feedback cycle of the learning system

lated in the error values of rules. The performance of the system will depend on the specification of the rule-selection function used in Step (2), the specification of the similarity measure in Step (4) and the method for evaluating rule error values in Step (5). In the simulations below, different specifications of the selection function (Step (2)) and methods for evaluating error values (Step (5)) are tested. Although the specification of a similarity measure also requires important operationalisation choices (see Joh et al. 2000, 2001; Wilson 1998), this component of the algorithm is kept constant in the simulations below. An Euclidean distance method is used, i.e. counting the number of mismatches between elements in corresponding positions in the activity strings (schedules). When needed, the lengths of the strings are made identical by adding 'missing' values to the end of the shortest string. Formally:

$$e(s,g) = \sum_{i=1}^{\min(l,m)} \vartheta_i - |l-m| \tag{3}$$

where

$$\vartheta_i = \begin{cases} 0 & \text{if } s_i = g_i \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

and $e(s, g)$ is the measure of similarity between strings s and g with elements s_i and g_i, and l and m are the length of s and g. As implied by this method, the maximum error value is zero (a perfect match between the strings) and the minimum value equals $-n$ (a mismatch for all the elements of the two strings), if the longest string contains n elements.

3.3 Calculating rule error values (Step (5))

Two methods for calculating error values are compared in the simulations below. In the first method, the error value of a rule is expressed as the

average error value across all the cases in which the rule delivered output. For clarity of presentation, we will drop the subscript t for the DT in the equations below. Then, the 'average-error' method can be written as:

$$\begin{aligned} e^0_{jk} &= 0 \\ e^h_{jk} &= \frac{\sum_{g=1}^{h} \delta_{jkg} e_g}{\sum_{g=1}^{h} \delta_{jkg}} \quad h > 0 \end{aligned} \tag{5}$$

where: e^h_{jk} is the error value of the k-th action alternative in the j-th column after processing h cases; δ_{jkg} is a binary variable denoting whether the k-th action alternative in the j-th column was selected to generate output in the g-th case and e_g is the measured error in reproducing the g-th case. In words, the values are initially set to zero and are updated each time a case is processed. In updating the value, the method assigns the full amount of error to each rule that contributed to the outcome. Hence, the system (falsely) assumes that each rule was fully responsible for the result. Because this assumption is not correct, the absolute values have no meaning. At best, the values give an indication of average rule performance across all the cases where the rule delivered output.

In the above 'average-error' method, the past performance of a rule keeps exerting influence on the current rule value throughout training. Although the effect of current performance on the running average becomes weaker and weaker as training proceeds, the system exhibits a complete memory of past cases. In the second method, the system still keeps record of past performance, but the weight of past cases on current error values declines as training proceeds. This 'limited-memory' method can be written as follows:

$$\begin{aligned} e^0_{jk} &= 0 \\ e^h_{jk} &= \beta e^{h-1}_{jk} + \delta_{jkh} e_h \quad h > 0 \end{aligned} \tag{6}$$

where $0 \leq \beta \leq 1$ is a pre-specified constant and other elements are defined as above. The equation has a recursive structure resulting in the following series:

$$\begin{aligned} e^0_{jk} &= 0 \\ e^1_{jk} &= \delta_{jk1} e_1 \\ e^2_{jk} &= \beta \delta_{jk1} e_1 + \delta_{jk2} e_2 \\ e^3_{jk} &= \beta^2 \delta_{jk1} e_1 + \beta \delta_{jk2} e_2 + \delta_{jk3} e_3 \\ e^4_{jk} &= \beta^3 \delta_{jk1} e_1 + \beta^2 \delta_{jk2} e_2 + \beta \delta_{jk3} e_3 + \delta_{jk4} e_4 \\ &\vdots \end{aligned} \tag{7}$$

That is, each time a case is processed a new error term is added to the right-hand side of the equation. The existing error terms reflect scores in previous cases and, thus, constitute the memory of the system. If $\beta < 1$, the weight of past cases in the current rule error value decreases reflecting an incomplete memory.

The strength of memory (i.e., the influence of history) depends on the chosen value of β. As the value approaches one, the strength of memory increases. Zero and one values correspond to the special cases of no memory and unlimited memory, respectively. Note that similar constructions have also been used in studies concerned with modelling history in consumer choice behaviour.

Both the 'average-error' and 'limited-memory' methods are incremental. The advantages of incremental learning are that the process is continuous, accepts noisy data and error values remain bounded.

3.4 The rule-selection function (Step (2))

Each time a case is supplied, the system selects for each column within DTs an action alternative based on comparing error values. The following alternative selection functions are investigated: (i) equal probability; (ii) linear probability; (iii) constant exponential probability and (iv) variable exponential probability.

First, the *equal* probability function is included as a null-model against which the performance of other functions can be evaluated. This function simply involves a random selection of alternatives. In the other functions ((ii), (iii) and (iv)) the selection probability is a function of weights which are derived from rule error values, as follows:

$$w_{jk} = \frac{\min_k e_{jk}}{e_{jk}} \tag{8}$$

where w_{jk} is the weight of the k-th action alternative in the j-th column. This equation applies only to non-zero distributions of e_{jk} across k. If one or more of the values equals zero, the system assigns a zero-one weight distribution to alternatives k. Since error values are equal to or smaller than zero, the above equation results in weights which are equal to one for the worst alternative and larger than one for the other alternatives. The *linear* probability function is defined as:

$$p_{jk} = \frac{w_{jk}}{\sum_k w_{jk}} \tag{9}$$

where p_{jk} is the probability of selecting the k-th alternative in the j-th column. The *constant exponential* probability function is defined as:

$$p_{jk} = \frac{\mu^{w_{jk}}}{\sum_k \mu^{w_{jk}}} \tag{10}$$

where $a > 1$ is a given constant. Finally, the *variable exponential* probability function is defined as:

$$p_{jk} = \frac{\left(1 + \frac{1}{\gamma} n\right)^{w_{jk}}}{\sum_k \left(1 + \frac{1}{\gamma} n\right)^{w_{jk}}} \tag{11}$$

where n is the number of cases that has been processed and $\gamma > 1$ is a given scale factor.

In the exponential functions 10 and 11, selection probability grows exponentially (rather than linearly) with increasing rule weight. Hence, compared to the linear function the exponential functions display a stronger preference for least-error rule alternatives. In the first function (Eq. 10), the base number μ is a given constant, whereas in the second function (Eq. 11) the base number increases with the amount of received training (processed cases). Thus, in the latter function, the slope of the function is relatively flat in the initial stage and becomes increasingly steeper as training proceeds.

4 Simulation results

4.1 The case

The learning algorithm was tested based on the scheduling problem described in Sect. 2. The simulation program that was used consists of two modules: (i) a module for simulating attribute data in terms of household and activity-program characteristics and (ii) a scheduling system. Training of the system involves N times simulating the attribute data and observed schedule of a case and measuring the distance between the observed and generated schedule.

Attribute data for each case are generated by drawing randomly a value for each attribute assuming uniform probability distributions and zero correlations between attributes. The first attribute simulates differences in preferences dependent on household type. Two arbitrary household types were distinguished referred to as Type I and Type II. The activity program consists of maximally 6 activities. The available time period for conducting the activities was arbitrarily set to 20 units. An activity program was constructed by sequentially adding an activity to an initially empty set and randomly determining an attribute profile for the added activity. The profile consisted of the following attributes:

– the activity is yes/no in-home;
– the activity is yes/no mandatory;
– the begin time of the activity (a natural number in the range [0, 19]);
– the flextime of the activity (0, 10 or 20 time units);
– the duration of the activity (1, 2, 4 or 8 time units).

The flextime category specifies the degree of flexibility in terms of an interval around the specified begin time (0 units, ± 10 units, and ± 20 units). Activities were added to the program until the maximum of 6 was reached or a next activity would not fit in the available time period of 20 units given the durations of the activities. Thus, the activity program varies across cases with respect to the number of activities, the nature of the activities and timing characteristics. The time-pressure on the schedule was measured as the sum of the durations of the activities (the median value appeared to be around 14 units).

The scheduling system corresponds to the system described in Sect. 2. Recall that the system consists of (i) a scheduling algorithm, (ii) rules for evaluating the feasibility of activity positions and (iii) decision rules (DTs)

for determining the priority of activities, evaluating schedule positions and re-scheduling. With respect to the second component, the system incorporates only rules for evaluating temporal constraints, i.e. determining whether an activity fits into a position given begin time, flex time and duration characteristics of the activities. When a given activity does not fit in any one of the available positions it was not included in the schedule.

The DTs used for ranking activities and ranking schedule positions are shown in Figs. 1 and 2. In both decision tables, the condition section specifies four condition states (contexts) as specific combinations of household type (I, II) and time pressure (Low: < 14 units, High: ≥ 14 units). The strategies for activity ranking as well as position ranking, listed in the action sections, are arbitrarily chosen. For example, the fifth strategy of DT1 states that activities are to be scheduled in the order of increasing degree of flextime and, for example, the fifth strategy of DT2 describes a preference for clustering out-of-home activities. The rules define deterministic rankings of activities and schedule positions in each step. Ties are arbitrarily solved by selecting the first activity in the supplied program with equal priority or the first position in the schedule with equal preference. A 'real' scheduling system would incorporate additional DTs specifying (conditional) additional strategies for solving ties. This extension would make the system more realistic, but would not change the structural features which are of concern here. Finally, for the same reason also the last step, i.e. re-scheduling, was left out of consideration.

Observed schedules were generated by applying the same scheduling system with a pre-specified selection of action alternatives. The arbitrarily chosen alternatives are marked with an X in the figures. Given these specifications, learning involves discovering the true action alternatives for each condition state of DT1 and DT2, i.e. finding the specification of the action space that was used in these tables to generate the observations. Note that the length of schedules may vary from case to case as a consequence of the fact that an activity is skipped if no feasible position exists. To give an indication: in a typical case, around 60% of the schedules contained 6 activities, 38% contained 5 activities and the remaining 2% contained 4 activities.

4.2 Results

Figures 4 and 5 graphically show training results for different specifications of the rule-selection function, when the 'average-error' method (AE-method) and the 'limited-error' method (LM-method) is used to update error values of rules, respectively. The x-axis represents the number of training cases processed and the y-axis the results of a validation test. Validation involves processing an additional set of $M = 50$ observations based on the least-error rules (action alternatives) within the columns of DT1 and DT2. Performance is expressed as the average error across the supplied cases in terms of the similarity measure. A zero value indicates that all observations were correctly predicted after training and a value of –6 represents the theoretical minimum score (all positions incorrectly predicted in all cases). Each time after processing 20 cases, training was interrupted for a validation test and, again continued to process a next set

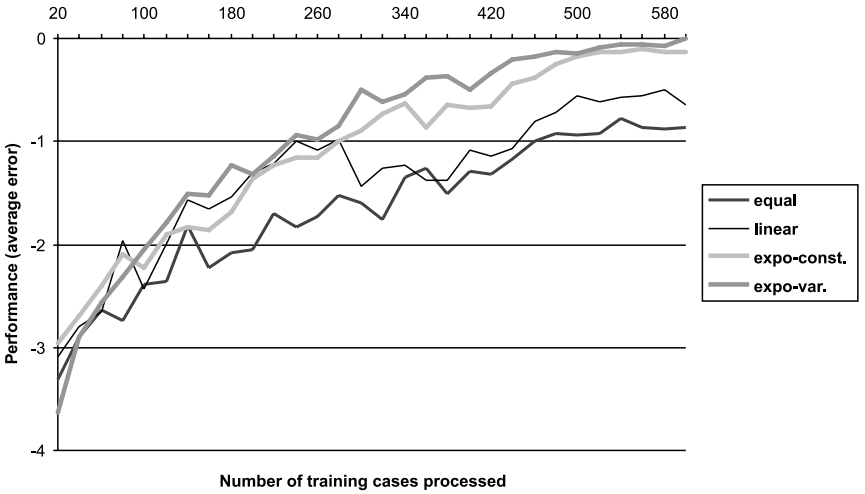

Fig. 4. Performance of the '*average-error*' rule-evaluation method under different rule-selection functions ($\mu = 20$, $\gamma = 25$)

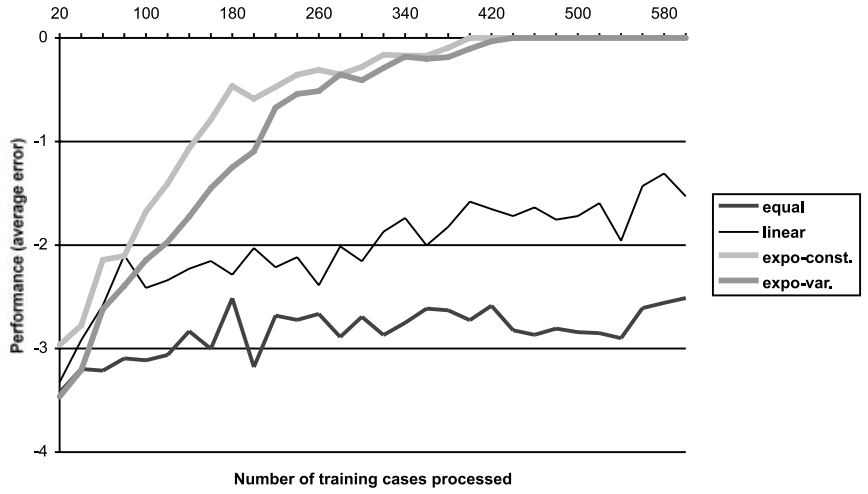

Fig. 5. Performance of the '*limited-memory*' rule-evaluation method under different rule-selection functions ($\mu = 20$, $\gamma = 25$, $\beta = 0.6$)

of 20 cases and so on. Thus, the figures show the continuous evolution of system performance during training. To obtain a clearer picture of trends, the data shown are averages across 5 runs of this combined training and validation procedure.

As a general finding, the system identified the 'true' rules in all runs and under a wide range of parameter settings both for the AE and LM-method. Figures 4 and 5 provide further details. First, with respect to the AE-method (Fig. 4), the system converged to the optimum solution in each variant of the

selection function except in the equal probability variant (i.e., random selection of rules). Speed of learning is considerably higher under the exponential functions compared to the linear variant. The constant and variable exponential functions approximately give the same results. In the case of the equal probability function, the system does not reach the optimum even if training is (considerably) prolonged. A closer look reveals that the system is able to identify the 'true' rules of DT2, but fails to find the 'true' rules of DT1.

In the case of the LM-method (Fig. 5), the same tendencies emerge but they are more pronounced. Here, the equal probability function does not result in visible learning effects. Under this null-model, the system even fails to identify the 'true' rules of DT2. The difference between the exponential and linear probability functions in learning speed is larger than in the AE-case. The constant and variable exponential functions perform more or less the same, like in the AE-case. Under optimal function specification conditions, the LM-method outperforms the AE-method in terms of learning speed.

It follows from these results that, in this case, the exponential functions outperform the linear alternative. Therefore, the (constant) exponential function was used in further tests of the robustness of the system for variations in parameter settings. Figure 6 shows the results when the base number, μ, is varied. As it turns out, the performance of the system is more or less constant for a wide range of parameter values. Figure 7 shows the result when the β-parameter in the LM-method is varied. Again, the system exhibits relatively constant performance within a wide range of this parameter. However, in the extreme case in which the parameter is zero (no memory), learning does not occur. Although under 'positive-memory' conditions differences are relatively small, there seems to be an optimum around $\beta = 0.6$.

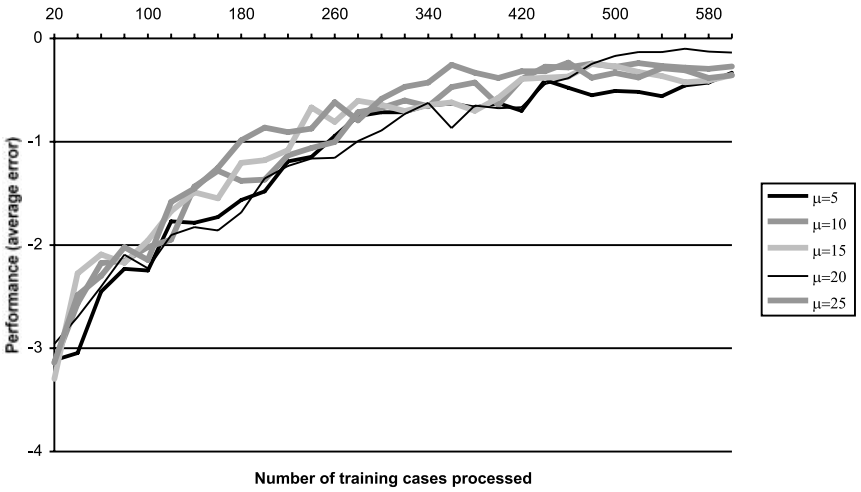

Fig. 6. Impact of the μ-parameter in the constant exponential selection function ('average-error' method)

Finally, Figs. 8 and 9 give insight in the underlying process in terms of the impact of training on least-error values within DT-columns in the AE and the LM-method, respectively. As an example, the data shown in the figures relate to DT1. Note that successful learning implies that least-error values approach zero as training proceeds (asymptotically in the case of the AE-method). In the AE-method the increase is more regular than in the LM-method ($\beta = 0.6$). In both cases, the system typically zooms in on rules with different speeds. This reflects the fact that the randomly generated training cases tend to be unequally distributed across the columns (i.e., combinations of household type and time pressure), particularly, in the earlier stages of the

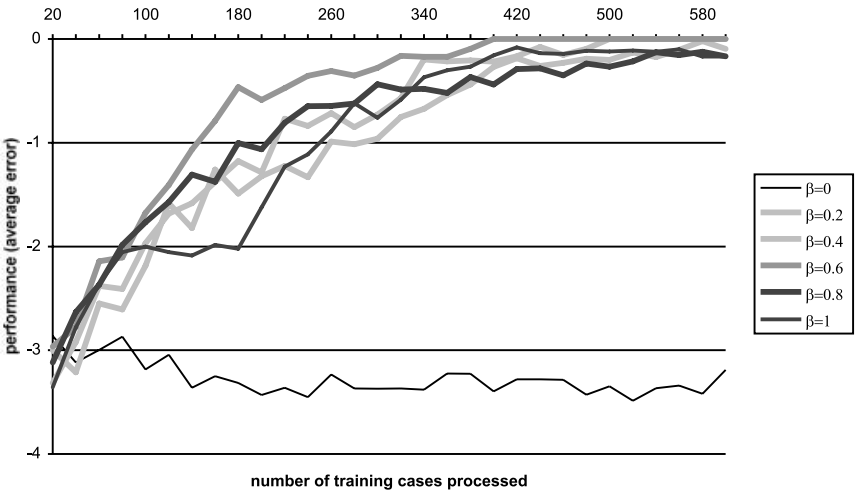

Fig. 7. Impact of the β-parameter in the 'limited-memory' method (constant exponential selection function, $\mu = 20$)

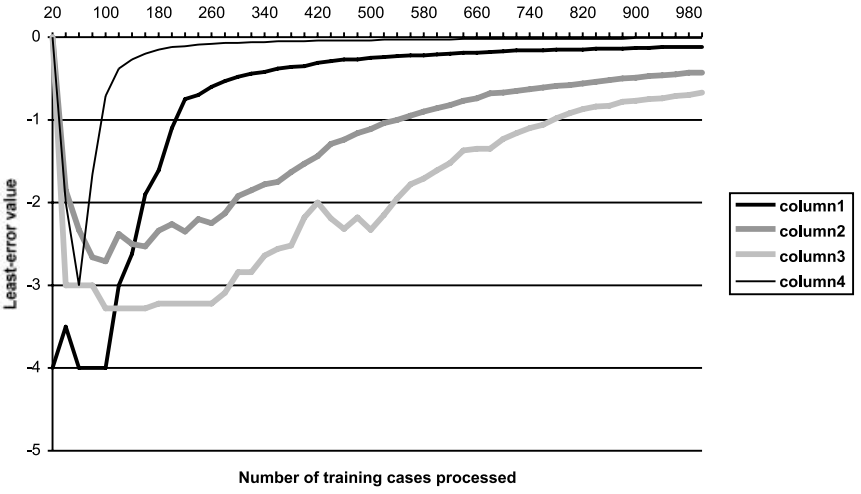

Fig. 8. Impact of training on least-error rule values per column of DT1 in the '*average-error*' method (constant exponential selection function, $\mu = 10$)

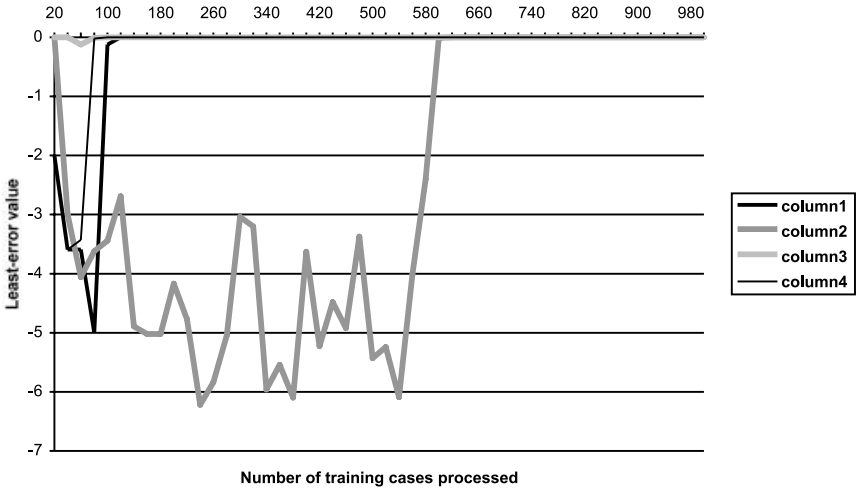

Fig. 9. Impact of training on least-error rule values per column of DT1 in the '*limited-memory*' method (constant exponential selection function, $\mu = 10$)

training process. In other words, the amount of received training at any moment in the process tends to differ across columns (competing rule-sets).

4.3 Interpretation of the results

The results suggest that, when properly specified, the system is capable of identifying combinations of rules that underlie a set of observations. An interpretation in terms of responsible mechanisms is important for generalising these findings to real-world cases. The poor performance of the equal-probability selection function reflects the existence of interactions between rules. As it turns out, the success of DT2-rules in ranking activity positions depends to some extent on the ranking of activities determined by the DT1-rules. Hence, the true DT2-rules are hard to identify under random selection conditions of DT1-rules and vice-versa. Linear and exponential rule-selection functions solve this problem through gradually assigning more weight to rules that prove to be successful. Initially, all rules are indifferent and, consequently, have equal probability of being selected. The rules that have independent positive effects on outcomes are the first to emerge. In subsequent cycles, these rules obtain higher priority and they create the conditions under which related rules can prove their value. In the next cycles, these dependent rules will become influential and, in turn, these rules may create the conditions for a third group of rules to emerge and so on.

In the present case, this process of progressively fixing strong rules always converged to an optimum solution. Compared to the linear function, the exponential functions are more sensitive to differences in error values and resulted in faster convergence. Further differentiation in sensitiveness through varying the slope parameter (μ or γ) of the exponential functions had only small impacts on learning speed. The case is not fully representative in the sense that real-world scheduling problems are characterised by more

decision dimensions, stronger interactions between rules and noisier feedback information. We may expect that in such more complicated cases the slope parameter becomes critical not only for the speed of learning but also for finding the optimum. Specifically, the μ-parameter of the constant exponential function has a theoretical optimum. With decreasing values the system may fail to identify weaker, but 'true' rules, whereas with increasing values the system may become too sensitive for non-systematic variance and is likely to display a tendency of selecting suboptimal paths.

In this respect, exponential functions with variable base number (γ) potentially combine the advantages of weak and strong sensitiveness. In these functions, the slope of the function becomes steeper as training proceeds. Initially, the system is relatively insensitive to differences in error values between alternative rules and, thus, reduces the risk of zooming in on a local optimum. As feedback information is accumulating, the system increasingly discriminates between alternatives and, thus, can optimally benefit from fixing strong rules. Comparable search mechanisms have also proven to be powerful in other combinatory-search systems (i.e., the annealing principle in optimisation methods).

Finally, there is an interaction with the strength of the memory of the system. In the 'limited-memory' method, memory strength can be varied continuously through the β parameter. Under strong memory conditions, the influence of past performance of rules on the probability of being selected is high and system's behaviour is more consistent. In the earlier stages of training, true rules are strongly under-evaluated as the assigned error values reflect performance under relative random rule-selection conditions. Hence, strong memory probably slows down convergence but, at the same time, improves the stability of the process. Again, in more complicated search spaces optimising the memory level will become more critical.

To conclude, the above mechanisms suggest that in more complicated cases the optimal specification of parameters controlling sensitiveness and memory strength of the system will vary from case to case dependent on characteristics such as consistency of feedback information, length of rule chains and strength of interactions between rules. Hence, the best strategy seems to be to fine-tune the system to the specific case at hand based on goodness-of-fit. The variable exponential selection function and the limited-memory method seem to be most flexible in this regard.

5 Conclusions and discussion

This paper introduced a learning algorithm for empirically deriving rule-based models of activity scheduling. For each scheduling facet decision, the conditional strategies individuals may possibly use are exhaustively represented in the form of a decision table. In learning from example cases, the system incrementally adjusts rule-error parameters determining probabilities of selecting rules for processing a next case. Various specifications of the rule selection and rule error update functions were investigated using simulated data. Computer experiments suggest that under a sufficiently wide range of parameter settings, the system is able to learn effectively how to reproduce observed schedules.

We view that the proposed algorithm brings the problem of how to empirically derive rule-based models of activity scheduling a step closer to a solution. The learning-based approach has several advantages especially in the context of activity-based modelling. *First*, the system is able to account for heterogeneity in a population. By including socio-economic variables as potential condition variables, the system optimises condition-action rules within the socio-economic segments defined. *Second*, the system designer has control over the generalisability and interpretability of the model. If action alternatives are specified in terms of behavioural strategies, the system is forced to explain cases in the same behavioural terms. The number of parameters to be estimated equals the total number of columns across decision tables. Both behavioural interpretability of rules and the relatively small number of parameters reduce the risk of overfitting. *Third*, estimation of the rule-based model tends to be relatively insensitive for outliers. Because an individual case can cause at most a minor adjustment of rule values, the system tends to converge on the set of rules that best fits the majority of cases. *Fourth*, the incremental nature of the learning system provides a method of preserving consistency of the model across multiple data collections separated in time. A new data set can simply be supplied to a trained system to improve the accuracy of predictions or, if preferences of individuals have changed, to adapt the system to the changes in behaviour. Speed of adaptation can be controlled by the user by means of the memory parameter of the system. Finally, by defining the feedback function alternatively in terms of a schedule utility function, the same system can be used for simulating long-term adaptive behaviour of individuals in response to changed environments. Specified in that way, the system would describe the gradual process of transition to new scheduling strategies that are better adapted to the changed environment.

Even though we may expect, for the reasons mentioned above, that the risk of overfitting in this type of model is relatively small, a test is still needed to obtain an indication of validity of the model in 'real' applications. This can be done by training the model on a subset of cases and testing the validity of the model on the remaining cases. In general, approaches for tackling the problem of overfitting in supervised learning systems have received considerable attention in neural network models (e.g., Fischer 2000). These approaches are potentially relevant for the present type of model as well. Finally, cross-validation methods are useful to reduce the loss of data for training.

As is true for supervised learning systems in general, the data needs of the model are relatively big, but, as the case study illustrated, do not exceed the size of activity diary data sets that are typically used for (estimating) activity-based models. Having said this, the size of typical data sets do restrict the number of condition states that can be distinguished. Obviously, the number of conditions rapidly increases with the number of socio-economic and context variables one wishes to consider for creating homogeneous groups of observations. An important problem for future research, therefore, is to extend the model to optimise the condition states simultaneously with the conditional selection of rules. Possibly, this can be done by incorporating heuristics for splitting columns of DTs, for example, that are available in existing decision tree induction systems. Then, the system would incrementally improve on the segmentation of the condition space and selection of

rules and would not implement more condition states than required for creating homogeneous groups.

Finally, the present study focused on the selection and sequencing of a given set of possible activities. Subsequent decisions related to choices of a location, mode and duration for each activity were left out of consideration. This is not a limitation of the system. It is possible to extend the specification of the system to cover the more encompassing scheduling problem simply by adding a decision table and appropriate constraint rules for each activity dimension and run the same learning algorithm on the extended set. Although the search space becomes larger, there is no principle restriction imposed by the learning algorithm on the number of decision steps in the scheduling model. Only, the similarity measure needs to be extended to measure the distance between a generated and observed pattern in multi-dimensional space. Recently, multi-dimensional methods of measuring similarity between activity patterns have become available so that this extension can be readily implemented (Joh et al. 2000, 2002).

References

Arentze TA, Timmermans HJP (2000) Albatross: *A Learning-Based Transportation Oriented Simulation System*. European Institute of Retailing and Service Studies, Eindhoven, The Netherlands

Arentze TA, Timmermans HJP (2001) Albatross – A Learning-Based Transportation Oriented Simulation System. *Transportation Research B* (forthcoming)

Arentze TA, Borgers AWJ, Timmermans HJP (1996) The integration of expertise knowledge in decision support systems for facility location planning. *Computers, Urban Environments and Systems* 19:227–247

Ben-Akiva ME, Bowman JL (1995) Activity-based disaggregate travel demand model system with daily activity schedules. Workshop on *Activity Based Approaches: Activity Scheduling and the Analysis of Activity Patterns*, Eindhoven, The Netherlands

Bhat C, Koppelman F (1999) Activity-based modeling of travel demand. In: Hall R (ed) *Handbook of Transportation Research*, Kluwer Academic Publisher, New York, pp 35–61

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) *Classification and Regression Trees*. Wadsworth, Belmont, CA

Cullen I, Godson V (1975) Urban Networks: The structure of activity patterns. *Progress in Planning* 4:1–96

Ettema DF, Timmermans HJP (1997) Theories and models of activity patterns. In: Ettema DF, Timmermans HJP (eds) *Activity-Based Approaches to Travel Analysis*. Elsevier Science, Oxford, pp 1–36

Ettema DF, Borgers AWJ, Timmermans HJP (1994) Using interactive computer experiments for identifying scheduling heuristics. *7th International Conference of the Association for Travel Behaviour Research*, Santiago, Chile

Ettema D, Daly A, Jong de G, Kroes E (1997) Towards an applied activity based travel demand model. *IABTR-Conference*, Austin, Texas

Fayyad UM, Shapiro GP, Smyth P, Uthurusamy R (1996) *Advances in knowledge discovery and data mining*. MIT Press, London

Fischer M (2000) Methodological challenges in neural spatial interaction modelling: The issue of model selection. In: Reggiani A (ed) *Spatial Economic Science: New Frontiers in Theory and Methodology*. Springer, Berlin, pp 89–101

Gärling T, Kwan M-P, Golledge RG (1994) Computational process modelling of travel activity scheduling. *Transportation Research B* 25:355–364

Gärling T, Brännäs K, Garvill J, Golledge RG, Gopal S, Holm E, Lindberg E (1989) Household activity scheduling. *Transport policy, management and Technology towards 2001: Selected*

Proceedings of the Fifth World Conference on Transport Research 4, Western Periodicals, Ventura, pp 235–248

Golledge RG, Kwan M-P, Gärling T (1994) Computational process model of household travel decisions using a geographic information system. *Papers in Regional Science* 73:99–118

Hägerstrand T (1970) What about People in Regional Science? *Papers of the Regional Science Association* 23:7–23

Hayes-Roth B, Hayes-Roth F (1979) A Cognitive model of planning. *Cognitive Science* 3:275–310

Joh C-H, Arentze TA, Timmermans HJP (2001) Multidimensional sequence alignment methods for activity-travel pattern analysis: A comparison of dynamic programming and genetic algorithms. *Geographical Analysis* 33:247–270.

Joh C-H, Arentze TA, Hofman F, Timmermans HJP (2002) Activity pattern similarity: A multidimensional alignment method *Transportation Research B*, (forthcoming)

Jones PM, Dix MC, Clarke MI, Heggie IG (1983) *Understanding Travel Behaviour*. Gower, Aldershot

Kitamura R (1984) A Model of daily time allocation to discretionary out-of-home activities and trips. *Transportation Research* 18B:255–266

Kitamura R, Fujii S (1996) Two computational process models of activity-travel behavior. Paper Presented at the Informs San Diego Conference, San Diego

Kitamura R, Yamamoto T, Fujii S (1996) A discrete-continuous analysis of time allocation to two types of discretionary activities which accounts for unobserved heterogeneity. In: Lesort J-B (ed) *Transportation and Traffic Theory*. Elsevier, Oxford, pp 431–453

Kwan M-P (1997) GISICAS: An Activity-based travel decision support system using a GIS-interfaced computational process model. In: Ettema DF, Timmermans HJP (eds) *Activity-Based Approaches to Travel Analysis*, Elsevier Science, Oxford, pp 263–282

Kwan M-P, Golledge RG (1997) Computational process modelling of disaggregate travel behaviour. In: Fischer MM, Getis A (eds) *Recent Developments in Spatial Analysis: Spatial Statistics, Behavioural Modelling and Computational Intelligence*. Springer, Berlin Heidelberg New York pp 236–252

Lenntorp B (1976) *Paths in Space-Time Environment: A Time Geographic Study of Possibilities of Individuals*. Lund: The Royal University of Lund, Department of Geography. [= *Lund Studies in Geography*, *Series B*, Human Geography, vol. no 44]

Lucardie GL (1994) Functional Object-Types as a Foundation of Complex Knowledge-Based Systems. Ph.D.-Dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands

Newell A, Simon HA (1972) *Human Problem Solving*. Prentice-Hall, Englewood Cliffs

Pendyala R, Kitamura R, Chen C, Pas E (1997) An activity-based micro-simulation analysis of transportation control measures. *Transport Policy* 4:183–192

Recker WW, McNally MG, Root GS (1986a) A model of complex travel behaviour: Part 1: Theoretical Development. *Transportation Research* 20A:307–318

Recker WW, McNally MG, Root GS (1986b) A model of complex travel behaviour: Part 2: An Operational Model. *Transportation Research* 20A:319–330

Quinlan JR (1993) *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo

Sutton RS, Barton AG (1998) *Reinforcement Learning: An Introduction*. MIT Press, London

Vanthienen J, Wets G (1994) From decision tables to expert system shells. *Data and Knowledge Engineering* 13:265–282

Vause M (1997) A rule-based model of activity scheduling behaviour. In: Ettema DF, Timmermans HJP (eds) *Activity-Based Approaches to Travel Analysis*. Elsevier Science, Oxford, pp 73–88

Wets G (1998) Decision Tables in Knowledge-Based Systems: Adding knowledge discovery and fuzzy concepts to the decision table formalism. Ph.D.-dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands

Wilson WC (1998) Activity pattern analysis using sequence alignment methods. *Environment and Planning A* 30:1017–1038