

An Activity-Based Scheduling Framework Incorporating Reinforcement  
Learning

Marlies Vanhulsel

January 26, 2010







To my husband and Sri Lankaner



## Acknowledgements

Dit werk zou ik niet tot een goed einde gebracht kunnen hebben zonder de steun en hulp van vele anderen. Daarom wil ik langs deze weg al die mensen bedanken voor hun al dan niet wetenschappelijke bijdragen.

Vooreerst wil ik de Universiteit Hasselt bedanken voor haar financiële en praktische steun. In de vier jaar die ik er heb doorgebracht in het kader van mijn doctoraat, heb ik veel kansen gekregen en heb ik enorm veel geleerd.

I also would like to thank the external members of the examination committee, Ann Nowé, Harry Timmermans and Julie Thompson, for having reviewed this dissertation.

Natuurlijk zijn er ook nog een heleboel andere personen die het werk aanzienlijk hebben verlicht. Ik denk daarbij eerst aan mijn (ex-)collega's op Imob. Jullie zijn met te veel om namen te noemen, maar ik zou jullie willen bedanken voor de aangename werksfeer. Actually, I want to say the same to my non-Dutch speaking colleagues at Imob. Thank you for having made this job easier, and thank you for the daily “joke-time”; it all began with Peanuts... Thanks for having made this time a wonderful time; I will miss all of you.

Verder wil ik een aparte paragraaf wijden aan mijn collega's en ondertussen ook vriendinnen Kelly, Carolien en Els. Jullie hebben ervoor gezorgd dat mijn periode bij Imob een onvergetelijke tijd geworden is. Ik kon steeds op jullie rekenen voor inspiratie, een luisterend oor, steun en advies. Bedankt voor alle middag- en fruitpauzes! Jullie hebben er geen idee van wat die voor mij hebben betekend. Hopelijk kan ik voor jullie ooit iets terug doen.

Bovendien wil ik ook mijn nieuwe collega's op VITO bedanken. Jullie hebben je uiterste best gedaan om ons (Hans, ik hoop dat je het niet erg vindt dat ik voor ons twee spreek?) vanaf dag één welkom te laten voelen. Jullie hebben steeds begrip gehad voor mijn situatie

## ACKNOWLEDGEMENTS

---

en hebben me meermaals een hart onder de riem gestoken. In de relatief korte periode dat ik nu op VITO werk, heb ik reeds veel bijgeleerd. Ik ben verder ook blij dat ik me nu eindelijk voor meer dan 100% kan richten op deze nieuwe uitdaging.

Ook buiten het werkterrein zijn er heel wat mensen die ik wil bedanken, met name mijn familie, schoonfamilie, de ladies en vrienden uit allerlei hoeken. Dank je wel voor de schouderklopjes en om te begrijpen waarom ik (weer maar eens) “vanavond niet mee kon gaan”, en vooral bedankt om me er af en toe aan te herinneren dat het leven meer is dan (mopperen over) het werk alleen. Bedankt voor de gezellige terrasjes, lekkere etentjes, fijne bezoekjes en vrolijke momenten. Speciale aandacht gaat hierbij naar mijn metekindjes, Rob en Kato, wie ik bij deze plechtig beloof dat ik vanaf nu meer tijd zal hebben om in het weekend te komen spelen. Daarnaast wil ik in het bijzonder mijn schoonvader bedanken voor zijn interesse in mijn werk: jammer dat je er op de valreep niet bij kon zijn.

Ik wil ook van de gelegenheid gebruik maken om mijn ouders extra te bedanken voor de kansen die ze mij gegeven hebben. Ik heb veel van jullie geleerd en jullie staan aan de basis van de toekomst die me wacht. Bovendien hebben jullie altijd met raad en daad voor me klaar gestaan, wat niet altijd even evident was en evenveel geapprecieerd werd. Dank je wel, mama, om je kostbare sabbatsperiode te onderbreken voor het nalezen van deze onbegrijpelijke verzameling woorden.

Last but not least, wil ik mijn echtgenoot Wim bedanken voor zijn onvoorwaardelijke steun. Zonder jou zou ik al lang geleden gestopt zijn met het afwerken van dit doctoraat. Het is een wonder hoe je me steeds wist te motiveren en het geduld bleef vinden om mijn grillen te trotseren (is dit het understatement van het jaar?). Bedankt voor die duizend-en-één kleine en minder kleine dingen. Vanaf nu is er weer meer tijd om verder te bouwen aan ons huisje-tuintje-kindje in welke volgorde dan ook.



# Abstract

While making decisions, governments and policy makers wish to be supported by models in order to estimate the impact of their decisions on society as a whole. Transportation models comprise a major example of such decision supporting models, as they are applied to monitor travel behaviour, to evaluate policy decisions, to assess the environmental influence of traffic, etc. Traditionally, transportation modelling highly concentrated on trip-based modelling.

Yet, recently activity-based modelling is gaining importance. This type of modelling assumes that travel patterns are the result of activity schedules that individuals execute in their attempt to achieve certain goals, taking into account individual needs, preferences, opportunities and constraints. Consequently, activity-based models aim at simulating the individual decision-making behaviour considering the distinct activity-travel related dimensions simultaneously. As such, an activity-based model predicts for each individual which activities to perform at which locations, when to start these activities and for how long and which transport modes are used in order to get to the desired locations. The resulting activity-travel sequences constitute the basis of the assignment of the individual routes to the transportation network, and as such estimating aggregate travel demand (Ettema & Timmermans, 1997a; Timmermans, 2000).

As a result, activity-based transportation models offer the opportunity of predicting travel demand more accurately as they provide a more profound insight into individual activity-travel behaviour. Furthermore these models are capable of estimating more realistically the impact of a policy on transportation related issues, for instance traffic safety, environmental pollution and land use patterns. Yet, the majority of such models is still quite static. This signifies that these models disregard the interaction between individual agents, as well the

effect of unforeseen events which occur in the course of the execution of the activity schedule (e.g. unexpected travel times) (Arentze *et al.*, 2005).

To this end, the present research contributes to the state-of-the-art of activity-based travel-demand modelling by presenting a framework to simulate activity-travel sequences, taking these requirements into account. For this purpose, the entire prediction process from pre-processing the data up to analysing the activity-travel sequences generated by the core scheduling engine is designed and tested in this dissertation.

To start with, the suitability of reinforcement learning to generate activity-travel patterns based on observed activity-travel diary data is explored. As the traditional reinforcement learning technique is not capable of learning efficiently in large state and action spaces with respect to memory and computational requirements on the one hand, and of generalizing based on infrequent visits of all state-action pairs on the other hand, the  $Q$ -learning technique used in most applications, is enhanced, by implementing an incremental regression tree function approximator. Furthermore, to incorporate the impact of interactions between the different aspects of activity-travel decisions (Gärling *et al.*, 1997; Joh *et al.*, 2002), multi-actor reinforcement learning is introduced.

Next, the data feeding the algorithm is examined. These data consist of observed activity-travel diaries on the one hand, and corresponding socio-demographic data on the other hand. Because people differ in their needs and preferences, while facing different opportunities and constraints, the observed activity-travel diaries expose a large variation. From this perspective, the predictive power of the scheduling algorithm can be increased by splitting the observed activity-travel sequences, which serve as input to the algorithm, into a number of clusters displaying similar activity-travel behaviour. For that reason, the current research presents a technique, founded on work conducted by Wilson (2008) - in which a multidimensional extension of the well-known sequence alignment method (Wilson, 1998a) is introduced -, which is capable of estimating the dissimilarity between sequences, considering the activity type as well as the relative positions of the locations in a sequence with regard to one another and the distances travelled between these locations.

Based on the dissimilarities between the observed patterns calculated by means of this technique, the observed data are divided into a number of groups. Subsequently, these clus-

---

ters are linked to the socio-demographic data so as to formulate socio-demographic profiles matching the clusters by means of a classification tree. The profiles can now be used to partition the synthetic population according to their socio-demographic attributes.

Thereafter, the current dissertation describes the actual core of the scheduling engine, which is composed of five decision modules, each of which incorporates a reinforcement learning system enhanced with a regression tree function approximator and connected to the subsequent module through multi-actor reinforcement learning. The first module determines the activity duration. In the second module the agent decides whether or not he wants to execute the next fixed activity (in this case sleeping or working) while the agent chooses which activity to perform in the third module. The fourth module is held responsible for selecting the distance band of the location where the agent wants to execute the selected activity. Finally, the agent fixes the travel mode to get to this location in the fifth module.

The decisions in these modules are all guided by reward functions which are calibrated on the observed activity-travel sequences. The design of the reward functions incorporated in these modules, as well as the functioning of each of the modules, is examined carefully. To speed up the learning process, one prototype agent for each cluster is trained first and the knowledge acquired by these prototype agents is used to initialize the individual agents corresponding to a member of the synthetic population thereafter.

Finally, as the main goal of the current research comprises formulating a framework to simulate activity-travel patterns within an activity-based travel-demand model, the performance - both with respect to the predictive power and the computational requirements - of the presented scheduling engine is assessed. To begin with, the execution time of the algorithm is investigated, pushing forward some adjustments to the program code to increase its efficiency. In addition, two conceptual improvements are advanced. Firstly, more prototype agents are included in the scheduling algorithm. These prototype agents no longer match the clusters of similar activity-travel patterns, but are linked to the (terminal) nodes of the decision tree, attaching socio-demographic profiles to each of these clusters. Consequently, the membership degrees in the nodes - which are determined by the distribution of the observed sequences belonging to that node over these clusters - now serve as weighting factors in the reward functions.

The second suggestion to enable scaling up the algorithm concentrates on converting the  $\epsilon$ -greedy action selection strategy into a softmax action selection strategy. Such action selection strategy assigns a probability of being selected to every action within the set of feasible actions based on the experienced  $Q$ -values. As a result, the agent can recognize a set of optimal actions and their corresponding probability, rather than selecting either the most optimal action (i.e. exploit) or picking an action uniformly randomly (i.e. explore) as is the case for the  $\epsilon$ -greedy action selection strategy. Because both suggestions introduce more variability into the predicted activity-travel patterns, it is no longer required to attune the reinforcement learning to each individual agent in the synthetic population.

After implementing these modifications, the reinforcement learning framework proves to be able to train the prototype agents in a time span of one hour (for ten prototype agents) and to generate one-day activity-travel sequences at a time resolution of five minutes for a population of five million agents in the course of approximately ten hours. This execution time can even be reduced when utilizing multiple parallel processors.

Concerning the predictive performance of this scheduling algorithm, the resulting activity-travel patterns are set side by side to a set of observed test sequences by means of a multidimensional sequence alignment method and descriptive statistics with regard to the simulated versus the observed activity types, duration and location. The outcome of these analyses shows that the algorithm is particularly suited to predict activity-travel behaviour based on observed activity-travel diaries.

## Samenvatting

Overheden en beleidsmakers wensen ondersteund te worden door modellen die de impact van hun beslissingen op de maatschappij weergeven. Belangrijke voorbeelden van zulke beslissingsondersteunende modellen zijn transportmodellen die verplaatsingsgedrag onderzoeken, beleidsbeslissingen evalueren en de milieu-invloeden van verkeer beoordelen, enz. Vroeger bestonden transportmodellen vaak uit tripgebaseerde modellen.

Echter, recentelijk worden deze modellen steeds vaker vervangen door activiteitengebaseerde modellen, die ervan uitgaan dat verplaatsingsgedrag afgeleid is van de activiteiten die individuen (willen) uitvoeren om zo bepaalde doelen te bereiken. In dit proces moet uiteraard rekening gehouden worden met hun behoeftes, voorkeuren, mogelijkheden en beperkingen. Vanuit dit perspectief streven activiteitengebaseerde modellen ernaar individuele beslissingsprocessen te simuleren die de verschillende dimensies van activiteitenverplaatsingspatronen gelijktijdig in aanmerking kunnen nemen. Op die manier trachten activiteitengebaseerde modellen voor elk individu te voorspellen welke activiteiten hij uitvoert, hoe lang deze activiteiten duren en wanneer en waar deze plaatsvinden en met welk vervoersmiddel hij deze gewenste locaties kan bereiken. De resulterende activiteitenverplaatsingssequenties vormen de basis om de individueel afgelegde routes op het transportnetwerk te projecteren, en om zo de vraag naar verplaatsingen op geaggregeerd niveau te schatten (Ettema & Timmermans, 1997a; Timmermans, 2000).

Activiteitengebaseerde transportmodellen het mogelijk om de vraag naar verplaatsingen nauwkeuriger te voorspellen omdat ze een dieper inzicht bieden in individueel activiteitenverplaatsingsgedrag. Bovendien kunnen deze modellen de impact van een beleidsmaatregel op transportgerelateerde kwesties, zoals verkeersveiligheid, milieuvervuiling en landgebruikpa-

tronen, realistischer beoordelen. Echter, de meerderheid van dit type modellen is nog steeds vrij statisch, wat inhoudt dat deze modellen geen rekening houden zowel met de interactie tussen individuele agenten, als met het effect van onverwachte gebeurtenissen die optreden terwijl een activiteitenplan uitgevoerd wordt (zoals onvoorziene verplaatsingstijden) (Arentze *et al.*, 2005).

In dit opzicht, draagt dit onderzoek bij aan de stand van zaken met betrekking tot activiteitengebaseerde verplaatsingsmodellen door een kader te schetsen waarin activiteitenverplaatsingspatronen gesimuleerd kunnen worden, rekening houdend met deze vereisten. Het gehele proces - gaande van het voorbereiden van de data tot het analyseren van de activiteitenverplaatsingssequenties, die de uitkomst vormen van het planningsalgoritme - wordt in deze thesis ontworpen en getest.

Om te beginnen wordt een algoritme gebaseerd op leren door bekrachtiging, ofwel reinforcement learning, bestudeerd en wordt de geschiktheid om activiteitenverplaatsingspatronen te genereren uitgaande van geobserveerde activiteitenverplaatsingsdagboekjes met behulp van zulk algoritme onderzocht. Hieruit blijkt dat het traditionele reinforcement learning algoritme niet efficiënt leert met betrekking tot geheugencapaciteit en rekenvereisten in probleemgebieden met een groot aantal toestanden (states) en acties (actions). Verder is het onmogelijk binnen het traditionele reinforcement learning algoritme om op basis van een klein aantal ervaringen van een onvolledige set toestanden en acties te veralgemenen voor alle toestanden en acties. Daarom stelt het huidige onderzoek voor om de veelvuldig gebruikte *Q*-learning techniek aan te vullen met een benadering door middel van een incrementeel regressieboomalgoritme. Om bovendien de wisselwerking tussen de verschillende aspecten van activiteitenverplaatsingsbeslissingen (Gärling *et al.*, 1997; Joh *et al.*, 2002) te kunnen modelleren, ontleent het huidige algoritme concepten van multi-actor reinforcement learning.

Daarnaast worden in deze thesis de inputdata van het algoritme onder de loep genomen. Deze data bestaan uit geobserveerde activiteitenverplaatsingsdagboekjes en socio-demografische gegevens. Aangezien mensen verschillen in hun behoeftes en voorkeuren terwijl ze rekening moeten houden met verschillende kansen en beperkingen, leggen de geobserveerde activiteitenverplaatsingsdagboekjes een grote verscheidenheid aan de dag. Uit dit oogpunt kan de voorspellingskracht van het planningsalgoritme verbeterd worden wanneer deze ge-

---

observeerde activiteitenverplaatsingssequenties onderverdeeld worden in groepen of clusters die verondersteld worden gelijkaardig activiteitenverplaatsingsgedrag te vertonen. Vanuit deze overweging wordt in het huidige onderzoek een multidimensionele uitbereiding van de bekende sequentie alignment methode (Wilson, 1998a) geïntroduceerd, die een maat definieert die aangeeft in hoeverre sequenties van elkaar verschillen met betrekking tot de activiteitstypes en de relatieve posities van de locaties binnen een sequentie en de afstanden afgelegd tussen deze locaties. Op basis van deze ongelijkheidsmaat kunnen de data opgedeeld worden in een aantal clusters. Vervolgens worden deze clusters gekoppeld aan de socio-demografische gegevens om zo te komen tot socio-demografische profielen die deze clusters beschrijven. Deze socio-demografische profielen kunnen aangewend worden om de synthetische populatie in te delen.

Vervolgens beschrijft de huidige scriptie de kern van het planningsalgoritme, dat bestaat uit vijf beslissingsmodules. Elk van deze modules is opgebouwd met een reinforcement learning systeem met een functiebenadering gebaseerd op regressiebomen en is verbonden met de daaropvolgende module met behulp van multi-actor reinforcement learning. De eerste module bepaalt de duur van elke activiteit. In de tweede module beslist de agent of hij de volgende vaste activiteit (d.i. slapen of werken) wil uitvoeren of niet. In de derde module selecteert de agent welke activiteit hij wil uitvoeren. De vierde module is verantwoordelijk voor het kiezen van een afstandsbereik voor de plaats waar deze activiteit moet plaatsvinden. Tot slot legt de agent het verplaatsingsmiddel om deze locatie te bereiken vast in de vijfde module.

De nutsfuncties die de beslissingen in deze modules ondersteunen, worden geïjkt op basis van de geobserveerde activiteitenverplaatsingssequenties. Het ontwerp van deze nutsfuncties en de werking van elke module worden onderworpen aan diepgaand onderzoek. Om het leerproces te versnellen, wordt eerst per cluster één prototypeagent getraind en de kennis die deze agenten opgedaan hebben, wordt vervolgens gebruikt om de individuele agenten - die overeenkomen met een individu uit de synthetische populatie - te initialiseren.

Tot slot wordt de werking van het voorgestelde algoritme met betrekking tot de voorspellingswaarde en de rekenvereisten geanalyseerd, aangezien het huidige onderzoek zich eerst en vooral richt op het formuleren van een kader om activiteitenverplaatsingspatronen te simule-

ren binnen een activiteitengebaseerd verplaatsingsmodel. Om te beginnen wordt de looptijd van het algoritme bestudeerd. Uit deze analyse komt naar voren dat een aantal aanpassingen aan de programmacode noodzakelijk zijn om de efficiëntie ervan te verhogen. Daarnaast worden twee conceptuele verbeteringen voorgesteld. Ten eerste worden meer prototypeagenten opgenomen in het planningsalgoritme. Deze prototypeagenten zijn niet langer gekoppeld de clusters met gelijkaardige sequenties, maar komen nu overeen met een (eind)knooppunt in de beslissingsboom die socio-demografische profielen definieert voor elke cluster. Om dit mogelijk te maken worden de lidmaatschapsgraden binnen deze knooppunten - die bepaald worden door de verdeling van de geobserveerde sequenties binnen het knooppunt over de verschillende clusters - nu gebruikt als wegingsfactoren in de nutsfuncties.

Het tweede voorstel om het algoritme te verbeteren legt de nadruk op de introductie van een softmax actieselectiestrategie in plaats van de eerder gebruikte  $\epsilon$ -greedy actieselectiestrategie. Deze actieselectiestrategie leidt uit de  $Q$ -waardes de kans af dat een actie, die behoort tot de set van mogelijke acties, geselecteerd wordt. Hierdoor erkent de agent een set van optimale acties en hun bijhorende kansverdeling in plaats van ofwel de meest optimale actie te selecteren (d.i. exploiteren) ofwel een actie willekeurig volgens een uniforme verdeling te kiezen (d.i. verkennen), zoals het geval is voor de  $\epsilon$ -greedy strategie. Beide suggesties hebben meer variabiliteit in de voorspelde activiteitenverplaatsingspatronen tot gevolg, waardoor het niet meer nodig is om de reinforcement learning modules op elke specifieke individuele agent binnen de synthetische populatie af te stemmen.

Na de implementatie van deze wijzigingen blijkt dat het reinforcement learning algoritme de prototypeagenten kan trainen in een tijdspanne van één uur (voor tien prototypeagenten) en dat het tien uur in beslag neemt om activiteitenverplaatsingspatronen te simuleren voor één dag voor een populatie bestaande uit vijf miljoen agenten en een tijdsresolutie van vijf minuten. Dit proces kan nog versneld worden wanneer er gewerkt wordt met verschillende parallelle processors.

Met het oog op het schatten van de voorspellingswaarde van dit planningsalgoritme worden de gesimuleerde activiteitenverplaatsingspatronen vergeleken met een set geobserveerde test sequenties met behulp van een multidimensionele sequentie alignment methode en een aantal beschrijvende statistieken op het niveau van de activiteitscategorieën, de activiteits-



---

duur en -locatie. Het resultaat van deze analyses toont aan dat het algoritme in het bijzonder geschikt is om activiteitenverplaatsingsgedrag te voorspellen op basis van geobserveerde activiteitenverplaatsingsdagboekjes.

SAMENVATTING

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Research Setting . . . . .	1
1.2	Travel-Demand Models . . . . .	2
1.3	Basic Requirements for Activity-Based Models . . . . .	5
1.4	Research questions . . . . .	8
1.5	Contributions . . . . .	9
1.6	Outline . . . . .	9
<b>2</b>	<b>Methodology</b>	<b>11</b>
2.1	Introduction . . . . .	11
2.2	Reinforcement Learning . . . . .	12
2.2.1	Fundamentals . . . . .	12
2.2.2	$Q$ -Learning . . . . .	14
2.2.3	Bucket-Brigade Updating . . . . .	17
2.2.4	Disadvantages of Reinforcement Learning Approach . . . . .	18
2.3	Function Approximation . . . . .	18
2.3.1	Regression Tree-Based Function Approximation . . . . .	19
2.3.2	Regression Tree Algorithm . . . . .	20
2.4	Multi-Actor Reinforcement Learning . . . . .	49
2.5	Related Research Efforts . . . . .	56
2.6	Conclusions . . . . .	57

## CONTENTS

---

<b>3</b>	<b>Data Pre-processing</b>	<b>59</b>
3.1	Introduction . . . . .	59
3.2	Data . . . . .	60
3.2.1	Data Collection Effort . . . . .	60
3.2.2	Some Descriptive Statistics . . . . .	61
3.3	Measures of (Dis)similarity . . . . .	69
3.3.1	Sequence Alignment Method . . . . .	69
3.3.2	Multidimensional Dissimilarity Measure . . . . .	73
3.3.3	Spatio-Temporal Dissimilarity Measure . . . . .	74
3.4	Identification of Groups of Similar Behaviour . . . . .	89
3.4.1	Clustering . . . . .	89
3.4.2	Validation of the Spatio-Temporal Dissimilarity Measure . . . . .	92
3.4.3	Design of Socio-Demographic Profiles . . . . .	100
3.5	Conclusions . . . . .	106
<b>4</b>	<b>Design of the System</b>	<b>109</b>
4.1	Introduction . . . . .	109
4.2	Module 1: Duration . . . . .	116
4.2.1	Reward Function . . . . .	116
4.2.2	Validation . . . . .	125
4.3	Module 2: Fixed Activities . . . . .	142
4.3.1	Reward Function . . . . .	142
4.4	Module 3: Activity selection . . . . .	144
4.4.1	Reward Function . . . . .	144
4.4.2	Validation . . . . .	148
4.5	Module 4: Location . . . . .	153
4.5.1	Reward Function . . . . .	153
4.5.2	Validation . . . . .	159
4.6	Module 5: Travel mode . . . . .	159
4.6.1	Reward Function . . . . .	159
4.6.2	Validation . . . . .	161

4.7	Conclusions . . . . .	164
<b>5</b>	<b>Performance</b>	<b>167</b>
5.1	Introduction . . . . .	167
5.2	Analysis of the Performance of the Algorithm . . . . .	168
5.2.1	Preliminary Operations . . . . .	168
5.2.2	A First Glance at the Performance of the Algorithm . . . . .	175
5.2.3	Streamlining of the Program Code . . . . .	177
5.3	Suggestions to Scaling Up the Algorithm . . . . .	178
5.3.1	Increase in the Number of Prototype Agents . . . . .	178
5.3.2	Softmax Action Selection . . . . .	180
5.4	Validation . . . . .	181
5.4.1	Computational requirements . . . . .	181
5.4.2	Predictive Power . . . . .	184
5.5	Conclusions . . . . .	201
<b>6</b>	<b>Final Conclusions</b>	<b>203</b>
6.1	Key Findings . . . . .	203
6.2	Topics for Further Research . . . . .	207
<b>A</b>	<b>Sequences in Short Form: Results</b>	<b>211</b>
A.1	Identification of Groups of Similar Behaviour . . . . .	211
A.1.1	Clustering . . . . .	211
A.1.2	Design of Socio-Demographical Profiles . . . . .	217

CONTENTS

---

## List of Tables

2.1	Reinforcement learning algorithm: $Q$ -learning . . . . .	16
2.2	Reinforcement learning algorithm: Bucket-Brigade updating . . . . .	17
2.3	Reinforcement learning algorithm: Regression tree-based function approxima- tion . . . . .	20
2.4	Parameter settings for model tree induction algorithms . . . . .	31
2.5	Impact of parameter $f$ on accuracy of the batch induction algorithm . . . . .	41
2.6	Impact of parameter $n$ on accuracy of the batch induction algorithm . . . . .	42
2.7	Impact of parameter $f$ on accuracy of the batch/incremental induction algorithm	43
2.8	Impact of parameter $n$ on accuracy of the batch/incremental induction algorithm	44
2.9	Comparison of tree induction algorithms . . . . .	48
2.10	Rewards assigned to choice of activity and location . . . . .	51
2.11	Multi-actor reinforcement learning process . . . . .	53
3.1	Descriptive statistics concerning the activity-travel sequences . . . . .	63
3.2	Average silhouette width for varying number of clusters $k$ based on the pro- posed spatio-temporal dissimilarity measure . . . . .	94
3.3	Average silhouette width for varying number of clusters $k$ based on the existing distance measure . . . . .	96
3.4	Descriptive statistics of cluster results based on the proposed spatio-temporal dissimilarity measure . . . . .	99
3.5	Descriptive statistics of cluster results based on the existing distance measure	100
3.6	$P$ -values of ANOVA tests . . . . .	101

LIST OF TABLES

---

3.7	Outcome of decision tree attaching socio-demographical profiles to the cluster results . . . . .	105
4.1	Parameters of the best curve fitted to the observed data of cluster 1 . . . . .	124
4.2	Parameters of the alternative reward functions based in module 1 on the observed data of cluster 1 . . . . .	125
4.3	Validation results for the duration module for cluster 1 . . . . .	127
4.4	Validation results for the duration module for cluster 2 . . . . .	128
4.5	Validation results for the duration module for cluster 3 . . . . .	129
4.6	Root of average squared difference between the predicted duration and the average observed duration for the corresponding activity calculated in the training dataset . . . . .	131
4.7	Root of average squared difference between the predicted duration and observed duration of the activity in the corresponding validation sequence . . .	132
4.8	Validation results for the duration module for cluster 1 based on traditional reinforcement learning agents . . . . .	135
4.9	Validation results for the duration module for cluster 2 based on traditional reinforcement learning agents . . . . .	136
4.10	Validation results for the duration module for cluster 3 based on traditional reinforcement learning agents . . . . .	137
4.11	Parameters of the best curve fitted to the observed data of cluster 1 . . . . .	147
4.12	Parameters for the alternative reward functions in module 3 based on the observed data of cluster 1 . . . . .	148
4.13	Validation results for the fixed activity selection module 2 . . . . .	150
4.14	Validation results for the activity selection modules 2 and 3 generated by means of the SAM-based reward function . . . . .	151
4.15	Validation results for the activity selection modules 2 and 3 generated by means of the history-based reward functions . . . . .	152
4.16	Validation results for the activity selection modules 2 and 3: relative frequency (%) of sequences containing the specified activity type . . . . .	153



4.17 Validation results for the activity selection modules 2 and 3: average number of episodes of the specified activity type within the sequences containing this activity type . . . . . 153

4.18 Parameters for the reward functions in module 4 based on the observed data of cluster 1 . . . . . 155

4.19 Parameters for the reward functions in module 4 based on the observed data of cluster 2 . . . . . 156

4.20 Parameters for the reward functions in module 4 based on the observed data of cluster 3 . . . . . 157

4.21 Validation results for the location module 4 . . . . . 160

4.22 Validation results for DP-SAM calculated based on the short format sequences simulated by the entire multi-actor reinforcement learning system . . . . . 162

4.23 Validation results for DP-SAM calculated based on the long format sequences simulated by the entire multi-actor reinforcement learning system . . . . . 163

5.1 Number of activity episodes observed for the old and new activity classification 169

5.2 Some general descriptive statistics of the clusters . . . . . 171

5.3 Percentage of sequences in clusters containing the listed activities at least once 171

5.4 Number of observed activity episodes and average and standard deviation of the duration of these activity episodes . . . . . 171

5.5 Number of sequences containing activities and average and standard deviation of the total duration of the activities within these sequences . . . . . 172

5.6 Percentage of sequences in which the listed distance bands corresponds to the farthest location reached . . . . . 172

5.7 Outcome of decision tree attaching socio-demographical profiles to the cluster results . . . . . 174

5.8 Time usage of functions for the multi-actor reinforcement learning scheduler incorporating  $Q$ -tables . . . . . 176

5.9 Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator . . . . . 176

LIST OF TABLES

---

5.10 Time usage of functions for the multi-actor reinforcement learning scheduler incorporating  $Q$ -tables after improving code efficiency . . . . . 178

5.11 Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator after improving code efficiency 179

5.12 Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator incorporating the suggested enhancements . . . . . 182

5.13 Membership degrees for set of test cases . . . . . 183

5.14 Validation results for DP-SAM calculated based on the long format sequences simulated by the multi-actor reinforcement learning system for four scenarios 186

5.15 Validation results for DP-SAM calculated based on the short format sequences simulated by the optimized multi-actor reinforcement learning system . . . . 187

5.16 Validation results for DP-SAM calculated based on the long format sequences simulated by the optimized multi-actor reinforcement learning system . . . . 187

5.17 Some general descriptive statistics of the test sequences . . . . . 190

5.18 Some general descriptive statistics of the simulated sequences . . . . . 190

5.19 Percentage of test sequences containing the listed activities at least once . . . 191

5.20 Percentage of simulated sequences containing the listed activities at least once 191

5.21 Number of activity episodes and average and standard deviation of the duration of these activity episodes in the test sequences . . . . . 192

5.22 Number of activity episodes and average and standard deviation of the duration of these activity episodes in the simulated sequences . . . . . 193

5.23 Number of test sequences containing activities and average and standard deviation of the total duration of the activities in these sequences . . . . . 194

5.24 Number of simulated sequences containing activities and average and standard deviation of the total duration of the activities in these sequences . . . . . 195

5.25 Percentage of test sequences in which the listed distance bands corresponds to the farthest location reached . . . . . 196

5.26 Percentage of simulated sequences in which the listed distance bands corresponds to the farthest location reached . . . . . 196

5.27 Refined socio-demographical profiles of prototype P10 . . . . .	198
5.28 Refined membership degrees for set of test cases matching prototype P10 . . .	199
5.29 Validation results for DP-SAM calculated based on the short format sequences simulated by the optimized multi-actor reinforcement learning system, refined for prototype P10 . . . . .	200
5.30 Validation results for DP-SAM calculated based on the long format sequences simulated by the optimized multi-actor reinforcement learning system, refined for prototype P10 . . . . .	200
A.1 Average silhouette width for varying number of clusters $k$ based on the pro- posed spatio-temporal dissimilarity measure . . . . .	212
A.2 Average silhouette width for varying number of clusters $k$ based on the existing distance measure . . . . .	212
A.3 Descriptive statistics of cluster results based on the proposed spatio-temporal dissimilarity measure . . . . .	215
A.4 Descriptive statistics of cluster results based on the existing distance measure	216
A.5 P-values of ANOVA tests for the sequences in the short form . . . . .	217
A.6 Outcome of decision tree attaching socio-demographical profiles to the cluster results for the sequences in the short form . . . . .	220

LIST OF TABLES

---

## List of Figures

2.1	Weight of concept . . . . .	30
2.2	Impact of parameter $f$ on accuracy of the batch induction algorithm estimated on data excluding noise . . . . .	33
2.3	Impact of parameter $f$ on accuracy of the batch induction algorithm estimated on data including noise . . . . .	34
2.4	Impact of parameter $n$ on accuracy of the batch induction algorithm estimated on data excluding noise . . . . .	35
2.5	Impact of parameter $n$ on accuracy of the batch induction algorithm estimated on data including noise . . . . .	36
2.6	Impact of parameter $f$ on accuracy of the batch/incremental induction algorithm estimated on data excluding noise . . . . .	37
2.7	Impact of parameter $f$ on accuracy of the batch/incremental induction algorithm estimated on data including noise . . . . .	38
2.8	Impact of parameter $n$ on accuracy of the batch/incremental induction algorithm estimated on data excluding noise . . . . .	39
2.9	Impact of parameter $n$ on accuracy of the batch/incremental induction algorithm estimated on data including noise . . . . .	40
2.10	Comparison of tree induction algorithms estimated based on data excluding noise . . . . .	46
2.11	Comparison of tree induction algorithms estimated based on data including noise . . . . .	47
2.12	Multi-actor reinforcement learning system . . . . .	50

LIST OF FIGURES

---

2.13	Learning scheme . . . . .	53
2.14	Evolution of optimal composite action . . . . .	55
3.1	Histograms concerning the socio-demographical data: age and gender . . . . .	65
3.2	Histograms concerning the socio-demographical data: marital status and number of children . . . . .	66
3.3	Histograms concerning the socio-demographical data: work schedule and number of working hours . . . . .	67
3.4	Histograms concerning the socio-demographical data: income category and day of the week . . . . .	68
3.5	Hägerstrand trajectories of a subset of the data projected on a map of the study area . . . . .	70
3.6	Examples of unidimensional sequences . . . . .	71
3.7	Empty $N \times M$ matrix for sequence alignment . . . . .	71
3.8	$N \times M$ matrix for sequence alignment in the course of the alignment process . . . . .	72
3.9	$N \times M$ matrix for sequence alignment at the end of the alignment process . . . . .	72
3.10	Examples of multidimensional sequences containing the activity type and $(x, y)$ -coordinates of the activity location . . . . .	73
3.11	Hägerstrand trajectories of example sequences . . . . .	75
3.12	Sequences: rotated, translated and mirrored . . . . .	77
3.13	Calculating the trajectory of AAL-pairs for the base sequence . . . . .	78
3.14	(Normalized) trajectories of angles of the sequences of figure 3.12 compared to the (normalized) trajectory of angles of the base sequence . . . . .	80
3.15	Mirrored trajectory of normalized angles of the mirror sequences of figure 3.12 compared to the trajectory of normalized angles of the base sequence . . . . .	81
3.16	Sequences 2: rotated and mirrored . . . . .	82
3.17	(Normalized) trajectories of angles of the sequences of figure 3.16 compared to the (normalized) trajectory of angles of the base sequence . . . . .	83
3.18	Corrected (normalized) trajectories of angles of the sequences of figure 3.16 compared to the (normalized) trajectory of angles of the base sequence . . . . .	85
3.19	Hägerstrand trajectory of one of the sequences . . . . .	87

---

3.20	Normalized AAL-trajectory of sequence displayed in figure 3.19 . . . . .	88
3.21	Sequence represented in the long form (time slots of 1 hour) vs. the short form for the example sequence $S$ of figure 3.10 . . . . .	91
3.22	Hägerstrand trajectories of sequences represented in the long form (time slots of 1 minute) vs. the short form for the sequence recorded in figure 3.19 . . . . .	91
3.23	Cluster results for varying number of clusters $k$ based on the proposed spatio-temporal dissimilarity measure . . . . .	93
3.24	Cluster results for varying number of clusters $k$ based on the existing distance measure . . . . .	95
3.25	Determining the SDD-value for an observed sequence . . . . .	97
3.26	Determining the SDE-value for an observed sequence . . . . .	98
3.27	Decision tree attaching socio-demographical profiles to the cluster results . . . . .	102
3.28	Overall standard deviation of predictor variable . . . . .	103
4.1	Agent-based micro simulation framework . . . . .	111
4.2	Reinforcement learning system . . . . .	112
4.3	An example of the scheduling process illustrated . . . . .	115
4.4	Traditional reward function in module 1 for the working activity of cluster 1 . . . . .	117
4.5	Reward function using progress estimators in module 1 for the working activity of cluster 1 . . . . .	118
4.6	Example of rewards assigned . . . . .	119
4.7	Alternative reward function using progress estimators in module 1 for the working activity of cluster 1 . . . . .	121
4.8	Exploration rate implemented in module 1 for the traditional reinforcement learning agents . . . . .	139
4.9	Exploration rate generated in the course of the learning process for module 1 for the reinforcement learning agents enhanced with regression tree function approximation . . . . .	141
4.10	Reward function for module 2 for the fixed working activity . . . . .	143
4.11	Alternative reward function in module 3 for the grocery shopping activity of cluster 1 . . . . .	149

LIST OF FIGURES

---

4.12 Reward function in module 5 for the working activity of cluster 1 . . . . . 161

A.1 Cluster results for varying number of clusters  $k$  based on the proposed spatio-temporal dissimilarity measure . . . . . 213

A.2 Cluster results for varying number of clusters  $k$  based on the existing distance measure . . . . . 214

A.3 Decision tree attaching socio-demographical profiles to the cluster results for the sequences in the short form . . . . . 218

A.4 Overall standard deviation of predictor variable for the sequences in the short form . . . . . 219



# Chapter 1

## Introduction

### 1.1 Research Setting

Mobility proves to be a driving force impacting economy, society and the environment. For instance, the socio-economic effects of a well-constructed transportation network include opening up areas in order to attract capital and people while creating employment, preventing social exclusion and enhancing liveability. In this sense, mobility offers the means to bridge geographical distances, while taking into account temporal restrictions (Hägerstrand, 1970). However, mobility also brings along a number of less favourable side effects. One of these comprises the fact that transportation consumes a major part of total commercial energy and produces a huge amount of emissions, which are known to have negative environmental and health implications.

In addition, transportation appears to be the cause of a number of social and economic issues, for example traffic unsafety and congestion, creating as such a cost to society. Therefore, in order to assess the impact of mobility on non-transport-related issues, such as air quality, as well as to evaluate the influence of transport- and non-transport-related policies on mobility, transport modelling emerges (Fried *et al.*, 1977; Shiftan & Surhbier, 2002; Shiftan *et al.*, 2003; Stead & Banister, 2001). After all, governments and policy makers wish to be supported by models in order to estimate the impact of their decisions on society as a whole. Within this scope, transportation models are developed.

## 1.2 Travel-Demand Models

For this purpose, many modelling approaches can be utilized. A well-known and widely used method to replicate and simulate behaviour, is the four-step modelling approach, a so-called trip-based model including a series of mathematical models calibrated on trip origin-destination data (McNally, 2008). Even though a behavioural resistance with regard to this type of modelling exists, this modelling approach is still very popular and applied very frequently due to its simplicity in calculations and limited need of computational requirements.

Yet, as already indicated, this type of modelling contains a number of limitations as formulated in McNally (2000). Most importantly, as trip-based models focus on individual trips, they disregard the basic principle that the demand for travel is derived from activity participation, and that trips and activities within a pattern are spatially and temporally related. Furthermore, four-step models neglect the behavioural foundation underlying travel behaviour, such as complex choice sets, which are limited by personal and interpersonal constraints, household dynamics and interrelationships between travel and activity participation.

However, the breakthrough of computational capacity and the growing insight that individual travel behaviour should constitute the basis of travel-demand models, initiated a shift away from these four-step models. Travel-demand modelling moved to tour-based modelling, which are assumed to enhance the behavioural realism by joining series of separate trips into tours beginning and ending at home or work. This type of modelling is founded on the idea that each tour contains one main goal, the so-called primary destination. The trip to and from this primary destination can be interrupted by a number of intermediate stops. To identify the primary destination, tour-based models hypothesize the existence of a hierarchy within the tour purposes, namely (1) mandatory activities (work or school), (2) maintenance activities (shop or pick-up/drop-off) and (3) discretionary activities (social, leisure, other). Although these models offer greater behavioural realism, enable a more precise representation of travel and are more suited to assess the impact of transportation demand management policies compared to the four-step models, a third type of models, activity-based models, recently gained more importance.

Rather than concentrating on the connection between the activities executed in the course

of the same home- or work-based tour, activity-based models focus on the relationship between all activities executed in the course of a day/week/month and on the interaction between household members. Five assumptions constitute the basis of these activity-based models and are summarized in the remainder of this paragraph.

Firstly, the demand for travel is a derived demand from the demand for activities (Jones, 1979). Jones (1979) believes that travel cannot be studied in isolation as it is part of a sequence of events which occur in space and time. As such, he fosters the idea of the existence of space-time prisms, in which travel enables a trade off between time and space. This assumption is also supported by Hägerstrand (1970). Consequently, travel is required if an individual wishes to participate in activities at a location different from his home location. This viewpoint implies that trips are merely a by-product of activity choices and are thus defined by the set of activities, the set of destinations offering suitable facilities for these activities and the characteristics of the transport system.

Secondly, Chapin (1974) states that activity patterns are the means by which humans satisfy their needs and wants. An activity pattern is thus the result of the interplay between the propensity and opportunity to engage in certain activities. On the one hand, the propensity to execute an activity is guided by a set of energizing factors, being motivations and thought ways, and by a set of constraining factors, being roles and individual characteristics. Chapin (1974) distinguishes two types of motivations for performing an activity: an activity can be driven either by subsistence needs, for instance sleeping, eating and health care, including activities providing income to meet these basic needs, for example working, or by culturally, socially and individually defined needs, such as social and leisure activities. On the other hand, the opportunity to participate in an activity depends on the perceived availability of facilities and services and on the perceived quality of these facilities and services. The choice to execute a certain activity is a function of one or more wants, a set of perceived and feasible alternatives for achieving these wants and the perceived cultural and social context for making this choice. In this perspective, performing a series of activities results in satisfaction and feedback.

Thirdly, instead of starting from the motivations and opportunities of people, Hägerstrand (1970) focuses on the constraints limiting human behaviour. First, he distinguishes

capability constraints which restrict the activities people can execute because of their biological constructions and/or the tools one can operate. Second, he lists coupling constraints, which define the timing and location of activities and the available infrastructure to execute these activities. From this viewpoint, he states that telecommunication allows people to engage in certain activities without losing time in transportation. Next, Hägerstrand (1970) discerns authority constraints, which are mainly spatial and include the idea that a number of space-time entities are under the control of a certain individual or group, e.g. a piece of land or a home. Combining these constraints, Hägerstrand (1970) comes to the definition of so-called space-time prisms, which reflect these interdependencies between time and space.

Furthermore, interrelating this idea and the previous one, Gärling *et al.* (1997) substantiate that different aspects of activity and travel decisions are interdependent. This notion is also supported by Joh *et al.* (2002) and signifies that the utility of one decision component is influenced by the outcome of another decision component. For instance, the activity type decision affects the utility of the subsequent destination choice, which influences the utility of the departure time, travel mode decision and route choice.

Fourthly, given the fact that the characteristics of the household influence the characteristics of the individuals' activity patterns to a great extent, and as such impacting travel-demand as well, the understanding of individual activity and travel decisions can be enhanced by incorporating interactions between household members within a travel-demand model (Jones *et al.*, 1983).

Timmermans (2006) also stresses the importance of including interactions between individuals, and in particular between household members. In this context, he defines three types of decisions that have to be considered when modelling household interactions. The first category consists of resource allocation and usage decisions which take into account the assignment of constrained resources, such as transport modes, on the household level. Second, a number of household tasks and activities have to be performed by only one household member, for example bring/get activities or daily shopping. This type of decisions is called task and time allocation decisions. On the contrary, joint activity participation decisions refer to activities which have to be executed jointly, for instance leisure or social activities. Though these interactions are widely recognized, Timmermans (2006) points out that the

majority of the existing activity-based modelling efforts focuses on maximizing individual utility functions, instead of maximizing household utility functions.

Finally, activity and travel decisions are affected to a large extent by past and anticipated future events (Bowman, 1995). After all, human behaviour is often impacted by habits or inertia, and responses to change are known to display lags and asymmetry.

Recapitulating these concepts, an activity-based travel-demand model thus aims at predicting which activity is executed, where, when, for how long and which transport mode is used to get to the desired location (Arentze & Timmermans, 2005a). As a result, activity-based travel-demand models offer the opportunity of approaching travel-demand more realistically and predicting it more accurately than traditional four-step models, as they provide a more profound insight into the daily individual activity-travel behaviour (Algers *et al.*, 2005; Kitamura, 1996).

Moreover, if implemented well, activity-based travel-demand models offer the opportunity of assessing the implications of non-transport policies or technological developments on travel, examining the impact of transport-related policies on non-travel issues and gaining a qualitative and quantitative insight of role of travel in people's lives (Jones, 1979). Furthermore, activity-based micro simulations provide numerous additional advantages, including predicting along a continuous time-axis instead of the traditional aggregated peak/off-peak estimations, the ability of realistically assessing the impact of travel-demand measures on individual activity-travel behaviour, and hence on travel-demand, the flexibility and versatility with respect to specific study objects and policy scenarios, the control on the accuracy by defining the desired level of spatial and temporal resolution and the comprehensibility as an evaluation tool (Kitamura, 1996).

### **1.3 Basic Requirements for Activity-Based Travel-Demand Models**

However, in order to comply with the expectations, activity-based models have to meet a number of requirements, as outlined by Pendyala & Bhat (2006) and reviewed here.

First, travel-demand models should be responsive to changes in land use, socio-economic and demographic characteristics, such as changes in population and employment totals, household distribution by zone, income, car ownership, household size, dwelling unit type,

number of children, employment distributions by zone, occupation per industry and per type, person distribution by age, employment status and gender.

Subsequently, travel-demand models should be able to account for changes in the characteristics of the multi-modal transport network to assess the impact on modal level of service attributes, such as distance, time and cost. This requirement implies being responsive to changes in highway network speeds, transit route frequencies, introduction of new facilities, new highway links, new transit stops and new bicycle and pedestrian facilities.

Thirdly, these models should enable implementing transportation policies, for instance pricing policies, policies aiming at encouraging alternate mode use or alternative work/school arrangements. Furthermore, the models should be able to consider the impact of new technologies and to account for changes to the spatial and temporal resolution.

Last but not least, activity-based modelling approaches should accommodate the emerging behavioural paradigms and concepts discussed above. These include resuming interdependencies and interactions (for instance modal, temporal and spatial interdependencies among trips in a chain and chains in a day, interdependencies in activity engagement across days and weeks, household interactions and interdependencies between residence and work/school locations), constraints and flexibility (for example modal, situational, institutional, household and personal constraints and flexibilities), a positive utility of travel, time use and activity pattern analysis (such as taking into account history dependency, effects of substitution and generation of in-home versus out-of-home activities, induced demand, travel efficiency and behaviour processes and decision rules).

The purpose of the current research covers designing a framework for generating activity-travel sequences within such dynamic activity-based travel-demand model. A prominent requirement of this framework includes the ability to incorporate both short term and long term dynamics (Goodwin *et al.*, 1990). The former dynamics occur within a day and refer to rescheduling due to the effect of preceding decisions or events on subsequent choices, and/or of later objectives on earlier decisions. An example is the occurrence of an unforeseen event in the course of the execution of an activity - either a negative or a positive delay - or unexpected travel times, resulting in a time-surplus or time-lack situation which triggers short-term adaptation (Arentze *et al.*, 2005; Goodwin *et al.*, 1990).

Long term dynamics cover the impact of experiences of previous actions on activity-travel patterns over a longer time frame. While conducting activities, individuals build up expectations and beliefs based on the outcomes of their behaviour. New experiences cause these expectations and beliefs to be updated, and behaviour to change accordingly. This type of dynamics is subject to a high degree of inertia, implying a slow response rate. For instance, long term dynamics can cause household relocation or a change in the car availability of the household (Arentze *et al.*, 2005; Arentze & Timmermans, 2005b; Goodwin *et al.*, 1990).

Numerous activity-based models exist, which attempt to predict travel demand. Three major methodologies can be distinguished within these models. The first category consists of utility-maximizing models, which suggest that individual decision-making behaviour aims at maximizing the overall utility. These models typically include a multinomial or nested logit model, covering the different decision aspects (Timmermans, 2001). Examples of utility-maximizing models are COBRA (Wang & Timmermans, 2000), the Day Activity Schedule Model (Bowman, 1998), PATRICIA (Borgers *et al.*, 2002) and STARCHILD (Recker *et al.*, 1983).

Computational process models constitute the second type of model. These models explicitly focus on the decision process shaping activity-travel patterns. Computational process models mainly comprise context-dependent choice heuristics to model the scheduling process. The techniques used in these models include amongst others decision trees, neural networks and discrete choice models (Timmermans, 2001). Examples of computational process models are ALBATROSS (Arentze & Timmermans, 2004), AMOS (Kitamura & Fujii, 1998), GIS-CAS (Kwan, 1997), RAP (Kulkarni & McNally, 2001) and SCHEDULER (Doherty *et al.*, 2002).

The last category of models contains hybrid models which use the concept of utility but which do not necessarily aim at maximizing it. These models rather concentrate on integrating utility-maximizing and computational process models (Timmermans, 2001). Examples of hybrid models are AURORA (Arentze *et al.*, 2005), PCATS-RUM (Fujii *et al.*, 1998) and SMASH (Ettema *et al.*, 1996).

The present research effort fits in with this last category of models, as it fixes its attention on modelling individual behaviour by means of reinforcement learning - a technique which

borrowing some ideas from reward (utility)-driven behaviour. It is assumed here that reinforcement learning is able to provide an adequate activity-based travel-demand framework for several reasons.

Firstly, reinforcement learning is inspired on the human way of learning through trial-and-error interactions with a dynamic environment (Kaelbling *et al.*, 1996). Long term learning, as discussed in the previous paragraph, can thus be incorporated because changes in the environment influencing activity-travel behaviour are reflected through changes in the reinforcement signal. If such changes persist, the activity-travel behaviour of the reinforcement learning agent is adapted accordingly.

Next, the technique does not require scheduling for a full day ahead, but instead it enables scheduling one activity after another while the schedule is being executed (as is discussed in chapter 4). This way, short term dynamics are being accounted for, because unforeseen events are included in the schedule right away. Furthermore, reinforcement learning is particularly suited to account for future events as it is able to take future (delayed) rewards into consideration (Sutton & Barto, 1998).

An additional advantage of  $Q$ -learning, which is a particular version of reinforcement learning elaborated on in section 2.2 and implemented here, includes the fact that the algorithm does not require a model of the environment (Sutton & Barto, 1998). Finally, aiming at incorporating interactions between decision aspects, multi-actor reinforcement learning provides a solution, as is described in section 2.4.

## 1.4 Research questions

Consequently, the main research question representing the focus of this thesis, can be formulated as follows:

*Is reinforcement learning able to constitute a solid basis for modelling individual activity-travel behaviour?*

This research question can be converted into a number of subquestions, guiding the current research effort.



1. (a) What are the aspects of reinforcement learning restricting its applicability in the current study area?  
(b) Which adaptations are required to meet these limitations?
2. To which extent is reinforcement learning able to account for interactions?
3. (a) Which type of data is required to serve as input?  
(b) To which extent do these data require pre-processing for the benefit of the reinforcement learning algorithm?
4. Can a conceptual framework incorporating reinforcement learning be defined which aims at simulating activity-travel sequences?
5. (a) To which extent is this reinforcement learning framework able to generate meaningful activity-travel sequences based on observed data?  
(b) To which extent is this reinforcement learning framework able to do so within an acceptable time frame for a given synthetic population?

## 1.5 Contributions

This thesis contributes to the state-of-the-art of activity-based travel-demand modelling by developing a scheduling framework inspired on the human way of learning. The founding technique is reinforcement learning, which is enhanced with a regression tree-based function approximator to meet the major shortcomings of the algorithm. Furthermore, this manuscript describes the entire scheduling process starting from handling the data in order to define socio-demographic profiles based on an improved (dis)similarity measure, calibrating the parameters of the system, running the proposed algorithm and processing and validating the resulting activity-travel patterns.

## 1.6 Outline

This manuscript is organized as follows: chapter 2 first introduces some of the terminology and concepts used within reinforcement learning, which is the algorithm that founds the

framework proposed in this project. This chapter discusses some drawbacks attached to this algorithm and, advances a technique, in particular a regression tree function approximator, to meet these objections. Furthermore, chapter 2 examines an extension of traditional reinforcement learning to enable including interactions between decision components. Next, chapter 3 elaborates on the data which serve as input to the scheduling algorithm. This chapter presents a method to calculate the (dis)similarity between activity-travel patterns of different individuals on different days of the week, aiming at identifying homogeneous groups of which the members display similar activity-travel behaviour in time and space. To end with, socio-demographic profiles are attached to these groups in chapter 3. Thereafter, chapter 4 describes the main scheduling engine, which consists of five interconnected modules, each of which is in charge of reproducing a different aspect of activity-travel behaviour. This chapter also illustrates and validates the functioning of each component of the scheduling engine. Subsequently, chapter 5 examines the applicability of the proposed algorithm within a large-scale activity-travel based framework. This chapter suggests and implements a number of measures which drive at scaling up the algorithm in order to improve its usability. To end with, chapter 6 summarizes the conclusions of the current dissertation and indicate some topics for further research.

## Chapter 2

# Methodology

### 2.1 Introduction

In order to model a dynamic system of activity-travel behaviour, the ability to learn through interaction with an uncertain and constrained environment should be incorporated (Arentze & Timmermans, 2003; 2005b). After all, such interaction leads to adaptation of behaviour, including within-day rescheduling or longer-term changes in preferences, and thus in behaviour (Arentze & Timmermans, 2003; 2005b; Charypar & Nagel, 2005). Therefore, the presented framework should be able to model continuously adapting activity-travel choices, which are formed by learning through interaction with the environment (Arentze & Timmermans, 2003).

To this end, a model based on reinforcement learning - which replicates the nature of human learning through trial-and-error interactions with a dynamic environment (Kaelbling *et al.*, 1996) - is developed in the current research. Reinforcement learning also entails a number of eye-catching advantages compared to existing statistical and data mining techniques. First, the reinforcement learning approach introduced here does not require a model of the environment to be present. Next, reinforcement learning enables taking into account delayed decisions and their value. Finally, reinforcement learning is particularly suited in goal-directed problem areas.

The remainder of this chapter first introduces the fundamentals of reinforcement learning and its well-known variant,  $Q$ -learning. After discussing some of the disadvantages of this

traditional technique, section 2.3 elaborates on function approximation based on regression tree induction to tackle the reported restrictions. This section examines some tree induction algorithms and their applicability within the described framework. Subsequently, to incorporate interactions between decision components, multi-actor reinforcement learning is described and evaluated in section 2.4. To end with, the application of reinforcement learning in related research efforts is summarized in section 2.5.

## 2.2 Reinforcement Learning

In order to grasp the core of the model applied in the present research effort, a concise description of reinforcement learning is included here. Yet for a detailed review of reinforcement learning, the reader is referred to Kaelbling *et al.* (1996) and Sutton & Barto (1998).

### 2.2.1 Fundamentals

Reinforcement learning corresponds to the nature of human learning through trial-and-error interactions with a dynamic environment (Kaelbling *et al.*, 1996). In order to comprehend this reinforcement learning problem more thoroughly following key concepts are explained first (Kaelbling *et al.*, 1996; Mitchell, 1997b; Sutton & Barto, 1998).

**Agent.** An agent is the decision-making unit being considered.

**State.** A state  $s$  is defined by a number of dimensions which describe the conditions of the environment which are observable to the agent. This perception of the environment is the first connection of the agent with its environment. A reinforcement learning system is composed of a finite set  $S$  of feasible states.

**Action.** An action refers to the decision that can be taken in a certain state, and provides the second connection between the environment and the agent. An action  $a$  is also described by a number of dimensions. In each state  $s$ , the agent can choose from a number of feasible actions, which are contained in the action set  $A(s)$ .

**Transition function.** Executing an action  $a$  in state  $s$  changes the state of the environment into the next state  $s'$ . The transition function  $\delta : S \times A \rightarrow S$  determines this transition

from state  $s$  to state  $s'$  through action  $a$  and is unknown to the agent.

**Reward function.** A reward function  $R : S \times A \rightarrow R$  defines the feedback that the agent receives from its environment while making decisions and executing actions. The reward can be compared to the concept of utility in conventional choice models. Additionally, the nature of the reward, which can be either immediate or delayed, and direct or indirect, depends on the goal of the reinforcement learning problem.

**Policy.** A policy  $\pi : S \rightarrow A$  defines which action the agent performs in each state  $s$ .

The transition function and the reward function can either be deterministic or non-deterministic. A deterministic function always produces the same outcome for a given state  $s$  and action  $a$ . For non-deterministic functions, several outcomes for a given state  $s$  and action  $a$  are feasible, and are randomly drawn from a probability distribution. In the present research, these reward and transition functions are assumed to be deterministic.

In reinforcement learning, an agent observes the state  $s$  of the environment and selects an action  $a$  to execute at each time step. The objective of the agent is to find a policy  $\pi$ , which maximizes the expected sum of rewards over time, which is denoted as follows (Kaelbling *et al.*, 1996; Mitchell, 1997b):

$$V^\pi(s_t) \equiv r_t + r_{t+1} + r_{t+2} + \dots \quad (2.1)$$

As a reinforcement learning agent not only takes into account immediate rewards - he is also susceptible to delayed rewards -, a discounting factor  $0 \leq \gamma \leq 1$  is introduced to determine this cumulative reward. With respect to this discounting factor,  $\gamma^i r$  reflects the present value of a reward  $r$  received  $i$  time steps into the future (Sutton & Barto, 1998). A value of  $\gamma$  close to 0 signifies that immediate rewards are considered to be more valuable compared to delayed rewards; whereas a value of  $\gamma$  close to 1 indicates that future rewards impact the value of an action more (Mitchell, 1997b). The value  $V^\pi(s)$  of the cumulative reward of an arbitrary policy  $\pi(s)$  starting in state  $s$  can now be defined as follows:

$$V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \equiv \sum_i \gamma^i r_{t+i}. \quad (2.2)$$

The optimal policy  $\pi^*$  maximizes the discounted cumulative reward. It equals:

$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}} V^\pi(s), \forall s. \quad (2.3)$$

The value corresponding to this optimal policy is denoted as  $V^*$ . Otherwise stated, the agent seeks to find the optimal policy by defining in each state  $s$  the action  $a$  which enables reaching  $V^*$ . This goal corresponds to determining the action in each state that maximizes the sum of the immediate reward  $r(s, a)$  and the value  $V^*(\delta(s, a))$  of the optimal policy of the subsequent state  $s'$  ( $=\delta(s, a)$ ), discounted by  $\gamma$ . For a state  $s$ , this can be written as follows (Kaelbling *et al.*, 1996; Mitchell, 1997a):

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} [r(s, a) + \gamma V^*(\delta(s, a))]. \quad (2.4)$$

### 2.2.2 Q-Learning

The learning approach described above requires a model of its environment in order to be able to derive the optimal policy, as perfect knowledge of the immediate reward function  $R$  and of the state transition function  $\delta$  is needed to estimate the value function  $V$  (Mitchell, 1997b). However, perfect knowledge of these functions is usually not prevalent in real world settings. To that purpose, Watkins (1989) introduces a novel value evaluation function  $Q$  to select the optimal policy without requiring a model of the environment to be present. The  $Q(s, a)$ -value denotes the expected utility of taking action  $a$  in state  $s$  and following a fixed policy thereafter (Kaelbling *et al.*, 1996; Mitchell, 1997b; Watkins, 1989). The  $Q(s, a)$ -value is defined as follows:

$$Q(s, a) \equiv r(s, a) + \gamma V^*(\delta(s, a)). \quad (2.5)$$

$r(s, a)$  is the immediate reward of executing action  $a$  in state  $s$ .  $V^*(\delta(s, a))$  is the value of the optimal policy in the state determined by  $\delta(s, a)$ . The optimal policy  $\pi^*$  can be rewritten in terms of  $Q(s, a)$ :

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q(s, a), \quad (2.6)$$

and thus:

$$Q(s, a) = r(s, a) + \gamma \left\{ \max_{a'} [Q(\delta(s, a), a')] \right\}. \quad (2.7)$$

The global optimal policy now consists of successively selecting an action  $a$  based on the local  $Q(s, a)$ -values for state  $s$  (Mitchell, 1997b). The  $Q(s, a)$ -values are stored in a look-up table of which each entry matches a particular  $(s, a)$ -pair. On each encounter with a (state,action)-pair  $(s, a)$ , the corresponding  $Q(s, a)$ -values are updated.

Additionally, a  $Q$ -learning agent has to be able to take into account previously gathered experience. To this purpose, a new model parameter, the step-size parameter or learning rate  $\alpha_{t+1}(s, a)$ , is introduced. The learning rate expresses the weight assigned to the currently calculated  $Q(s, a)$ -value ( $r(s, a) + \gamma \max_{a'} [\hat{Q}_t(s', a')]$ ) compared to the  $Q_t(s, a)$  value, calculated and stored during a previous visit to the  $(s, a)$ -pair. The estimate  $\hat{Q}_{t+1}(s, a)$  can be calculated according to following formula:

$$\hat{Q}_{t+1}(s, a) \leftarrow [1 - \alpha_{t+1}(s, a)] \hat{Q}_t(s, a) + \alpha_{t+1}(s, a) \left\{ r(s, a) + \gamma \max_{a'} [\hat{Q}_t(s', a')] \right\}. \quad (2.8)$$

Consequently, the  $\hat{Q}_{t+1}(s, a)$ -value represents a weighted average of all experiences. The  $\hat{Q}_{t+1}(s, a)$ -values are proved to converge to the optimal  $Q^*(s, a)$ -values, provided that action  $a$  in state  $s$  is selected an infinite number of times and  $\alpha_{t+1}(s, a)$  decreases appropriately, i.e. according to following conditions (Kaelbling *et al.*, 1996; Mitchell, 1997b; Watkins, 1989; Watkins & Dayan, 1992):

$$\sum_t \alpha_{t+1}(s, a) = \infty. \quad (2.9)$$

$$\sum_t [\alpha_{t+1}(s, a)]^2 < \infty. \quad (2.10)$$

The estimation of the  $\hat{Q}_{t+1}(s, a)$ -values in the  $Q$ -learning algorithm is described in table 2.1. To start with, the  $Q$ -learning algorithm initializes all entries of the  $Q$ -table. Subsequently, the algorithm starts learning the optimal policy in the course of subsequent learning

episodes, in which the agent observes its current state  $s$  of the environment, selects an action  $a$  to perform, receives an immediate reward  $r(s, a)$  and calculates and updates the entry of the  $Q$ -table corresponding to the  $(s, a)$ -pair according to Equation 2.8 (Kaelbling *et al.*, 1996; Sutton & Barto, 1998; Watkins & Dayan, 1992).

---

Initialize each entry  $\hat{Q}(s, a)$  in the  $Q$ -table.  
Repeat:  
  Observe state  $s$ .  
  Select and execute action  $a$ .  
  Observe next state  $s'$ .  
  Observe reward  $r(s, a)$ .  
  Calculate  $\hat{Q}_{t+1}(s, a)$  based on  $(s, a, r(s, a))$ -triplet according to Equation 2.8.  
  Update  $Q$ -table.

---

Table 2.1: Reinforcement learning algorithm:  $Q$ -learning

Faced with a decision in each state, the agent has to decide whether to exploit previously gathered knowledge or to explore the possible actions. On the one hand, exploiting signifies choosing the action that is known to yield the highest reward. By doing so, the agent aims at reaching the state that is close to the currently best solution. This is a so-called greedy approach. Exploring, on the other hand, denotes selecting an action randomly from the set of possible actions. Exploring aspires to arrive at a state that might not be visited otherwise, and which may produce a higher reward than that of the most optimal action so far (Mitchell, 1997b; Sutton & Barto, 1998).

Several strategies exist to deal with this trade-off between exploration and exploitation, for instance the greedy strategy, randomized strategies and the interval-based techniques (Kaelbling *et al.*, 1996). In this research, the frequently used  $\epsilon$ -greedy strategy, which is a randomized strategy including greedy action selection, is incorporated to guide this trade-off. The parameter  $\epsilon$  reflects the probability of performing a non-greedy, random action selection instead of greedily selecting the optimal action. In the case of a  $\epsilon$ -greedy action selection strategy, “random” signifies that the action is selected randomly from all available actions according to a uniform distribution and independently from the action values.

Furthermore, as exploration generally occurs more in the beginning of the learning process - at that moment the agent still has to “discover” his possibilities in the environment -, the



algorithm starts with a rather large value of  $\epsilon$  and decreases this value in the course of the learning phase (Kaelbling *et al.*, 1996; Sutton & Barto, 1998).

### 2.2.3 Bucket-Brigade Updating

Although the order in which the  $\hat{Q}_{t+1}(s, a)$ -values are updated based on the experienced  $(s, a, r(s, a))$ -triplets, does not endanger the final convergence to the optimal policy, this order does influence the training efficiency to a large extent (Mitchell, 1997b). In particular, the convergence to the optimal policy can be somewhat accelerated by using so-called Bucket-Brigade updating. This approach does not compute and update the  $\hat{Q}_{t+1}(s, a)$ -values every time the agent executes the selected action, receives the corresponding reward and observes the next state. Instead, the  $(s, a, r(s, a))$ -triplets are stored in-between, and the  $\hat{Q}_{t+1}(s, a)$ -values of the visited  $(s, a)$ -pairs are calculated in reverse chronological order at the end of the learning episode. This way, the effects of delayed rewards are incorporated into the  $\hat{Q}_{t+1}(s, a)$ -value within the same learning episode. As a result, a lower number of learning episodes is required to learn a good approximation of the  $Q(s, a)$ -value (Driessens, 2004; Mitchell, 1997b). The reinforcement learning algorithm incorporating Bucket-Brigade updating is summarized in table 2.2.

---

Initialize each entry $\hat{Q}(s, a)$ in the $Q$ -table.
Repeat:
Repeat until the end of the learning episode:
Observe state $s$ .
Select and execute action $a$ .
Observe next state $s'$ .
Observe reward $r(s, a)$ .
Store $(s, a, r(s, a))$ -triplet in temporary table.
Repeat in reverse chronological order:
Calculate $\hat{Q}_{t+1}(s, a)$ based on stored $(s, a, r(s, a))$ -triplet according to Equation 2.8.
Update $Q$ -table.
Delete $(s, a, r(s, a))$ -entry from temporary table.

---

Table 2.2: Reinforcement learning algorithm: Bucket-Brigade updating

### 2.2.4 Disadvantages of Reinforcement Learning Approach

#### Curse of dimensionality

The  $Q$ -learning algorithm described in section 2.2.2, requires storing  $Q(s, a)$ -values of all feasible  $(s, a)$ -pairs in a look-up table. Furthermore, in order to converge to the optimal sequence, the  $Q$ -learning approach requires that each  $(s, a)$ -pair is visited at least once and preferably an infinite number of times during the training process (Mitchell, 1997b; Sutton & Barto, 1998). Therefore, this algorithm is only applicable to small state-action problems, due to the fact that the look-up table grows exponentially with the dimensionality of the state and action spaces. Large space problems thus require both a huge amount of memory to store the large  $Q$ -tables and a huge amount of time and data to estimate the  $Q(s, a)$ -values accurately (Sutton & Barto, 1998).

#### Limited Applicability

Moreover, it is not realistic to assume that an agent visits every feasible  $(s, a)$ -pair in the course of the learning process. Consequently, the algorithm has to be altered in order to enable using experience of only a limited subset of the state-action space to represent all  $(s, a)$ -pairs, even the ones that have never been visited (Sutton & Barto, 1998).

## 2.3 Function Approximation

Some approaches tackling these limitations, involve generalizing either state or action space. To this end, one can apply variable resolution discretization, which consist of either state or action aggregation. States or actions are joined together to reduce the resolution of the state and the action spaces. However, a bad discretization may introduce a hidden state in the problem area, whereas a too fine discretization does not solve the issue of the large amount of training data required to learn the optimal policy (Smart & Kaelbling, 2000; Uther & Veloso, 1998).

A better solution is to replace the discrete look-up tables - mapping the state and action spaces to a  $Q$ -function, i.e.  $S \times A \rightarrow Q$  - by function approximators capable of generalizing across similar states and actions to reduce these state and action spaces. For the purpose

of function approximation, existing generalization techniques from the area of supervised learning can be used (Kaelbling *et al.*, 1996; Smart & Kaelbling, 2000; Sutton & Barto, 1998).

### 2.3.1 Regression Tree-Based Function Approximation

Supervised learning is different from the core of reinforcement learning as it requires training data, which consist of examples provided from a knowledgeable supervisor outside the agent, in order to generalize to unseen situations. However, within the reinforcement learning technique, supervised learning offers the generalization required to enhance the performance of the core algorithm, based on examples gathered in the course of subsequent learning episodes. Supervised learning techniques include, amongst others, artificial neural networks, statistical curve fitting, pattern recognition, fuzzy logic and nearest neighbour methods (Kaelbling *et al.*, 1996; Sutton & Barto, 1998).

Most researches incorporating function approximation in reinforcement learning focus on generalizing either over the state space or over the action space so as to reduce the number of feasible state or action dimensions (Kaelbling *et al.*, 1996; Sutton & Barto, 1998). As opposed to this approach, the goal of the function approximation here is to generalize over the state and the action space simultaneously. Therefore, the current research proposes to estimate the  $\hat{Q}_{t+1}(s, a)$ -values based on experienced  $(s, a, r(s, a))$ -triplets based on tree induction. Well-known algorithms for tree induction include CART (Classification and Regression Tree) (Breiman *et al.*, 1984), ID3 (Mitchell, 1997a) and its extension, C4.5 (Quinlan, 1993).

A regression tree included in the reinforcement learning algorithm - called the  $Q$ -tree - provides the desired conversion of the combinations of the state and the action spaces  $S \times A$  to combinations of regions of the state and action spaces, denoted as  $\mathbf{S} \times \mathbf{A}$ . The  $Q$ -tree thus replaces the  $\hat{Q}$ -estimates belonging to the  $S \times A \rightarrow Q$ -mapping stored in the  $Q$ -table, by  $\hat{Q}$ -estimates derived from the  $\mathbf{S} \times \mathbf{A} \rightarrow Q$ -mapping which are saved in the leaves of the  $Q$ -tree.

To incorporate the use of a regression tree algorithm in the  $Q$ -learning approach, the traditional  $Q$ -learning algorithm is altered. A schematic overview of this extended  $Q$ -learning approach is illustrated in table 2.3. This technique resembles the traditional  $Q$ -learning

algorithm to a great extent.

---

```
Initialize  $Q$ -tree.  
Repeat:  
  Observe state  $s$ .  
  Select and execute action  $a$ .  
  Observe next state  $s'$ .  
  Observe reward  $r(s, a)$ .  
  Calculate  $\hat{Q}_{t+1}(s, a)$ -value based on  $(s, a, r(s, a))$ -triplet  
  according to Equation 2.8.  
  Update  $Q$ -tree.
```

---

Table 2.3: Reinforcement learning algorithm: Regression tree-based function approximation

Initially, the  $Q$ -tree only includes the root node. This implies that all states are represented by one state region and that all actions are merged to one action region. In the course of the successive learning episodes, which consist of observing the state  $s$  of the environment, selecting and executing an action  $a$ , observing the change in the state of the environment  $s'$  and receiving the reward  $r(s, a)$  attached to the  $(s, a)$ -pair, the algorithm collects the instances, composed of the values of the state and action dimensions and the corresponding  $\hat{Q}_{t+1}(s, a)$ -value, used to train the  $Q$ -tree. Based on the information contained in these instances, the algorithm splits the nodes of the  $Q$ -tree so as to distinguish between values of the state and action dimensions, which generate distinct  $\hat{Q}$ -values.

When selecting the best action - i.e. the action matching the highest attainable  $\hat{Q}$ -value for a given state  $s$  -, the reinforcement agent walks through the  $Q$ -tree only once to populate a list of action regions and their corresponding  $\hat{Q}$ -values. Next, the agent searches this list for the best attainable  $\hat{Q}$ -value and deduces a corresponding value for the action dimension. If more than one action can be defined for this action region, an action is selected randomly within the boundaries of the region.

### 2.3.2 Regression Tree Algorithm

#### Regression Tree Induction: Some General Concepts

Decision tree induction offers many advantages with respect to traditional statistical analyses. First, as opposed to most statistical techniques, the outcome of decision tree induction algorithms is simple to interpret: the resulting decision tree can easily be transformed into

a set of if-then rules (Witten & Frank, 2000). Furthermore, decision tree induction algorithms are well suited to handle large datasets composed of attributes of mixed data types (categorical or numerical) (Breiman *et al.*, 1984).

In addition, decision trees supply a nice way to deal with missing data as these values are simply treated as another possible value of the attribute (Witten & Frank, 2000). Moreover, decision tree induction algorithms are non-parametric or distribution-free. This implies for instance that the method does not require data that are normally distributed. In addition to this, outliers and noisy data, which affect the performance of statistical models negatively, do not harm the outcome of decision trees (Breiman *et al.*, 1984). The terminology and concept of decision tree induction are introduced in the remainder of this paragraph.

A decision tree consists of a number of nodes, which can be either interior or terminal, containing a subset of the set of training instances. An interior node splits this set into two or more sets according to a splitting rule. The resulting nodes are called child nodes of this parent node. The splitting rule consists of one of the independent attributes and a corresponding test value. The root node is the starting point of the decision tree, which holds all instances of the training set. A terminal node, or leaf, assigns a possible value to the dependent variable based on the instances assigned to this node and is not split further.

Decision tree induction aims at improving the overall accuracy of the prediction in a node by dividing the instances into child nodes according to an independent variable obtaining more homogeneous groups (branches) with respect to the dependent variable. A decision tree initially consists of a single node, the root node, containing all training instances. The regression tree is then refined by recursively adding splits to the existing interior nodes, according to the most optimal splitting rule. The resulting child nodes are split further until a stopping criterion to limit the growth of the tree and to prevent over-fitting the training data, is met with. The selection of the splitting attribute and the corresponding test value(s) as well as the possible number of children and the stopping criterion depend on the tree induction method used.

In data mining, two types of decision trees exist: classification and model trees. If the predicted value is categorical, classification trees are applied. On the other hand, model trees are used if the dependent variable is numeric; each leaf node of a model tree contains a

model such as a linear regression model (Witten & Frank, 2000). A particular case of such model tree is the so-called regression tree, in which the model of the dependent variable in each leaf equals the average value of this dependent variable of all instances belonging to that leaf.

Although several regression tree induction approaches exist, the current research imposes a number of requirements. First, the decision tree applied here has to be able to handle a *numeric dependent variable*, as the target variable of the tree is the continuous  $Q(s, a)$ -value. Therefore in the current research, a model tree induction technique is applied. Furthermore, as the data used to estimate the decision tree are subject to a continuously changing environment and preferences, the technique has to be *responsive to structural changes* on the one hand, while being insusceptible to temporary fluctuations - in this case noise originating from imperfect observations of the environment and rewards - on the other hand. Additionally, the regression tree has to be estimated based on a *continuous stream of data*. Moreover, the regression tree is fitted on a *large amount of data*, all of which cannot be stored due to memory limitations and ageing.

A regression tree induction algorithm which complies with these conditions, is described into detail in following paragraphs.

### **Batch Regression Tree Algorithm**

The goal of this section comprises providing a general introduction to the selected regression tree technique. The dependent variable of an instance  $i$  is denoted  $y_i$  and the independent variable or regressor  $j$  matching the  $i^{th}$  instance is labelled  $x_{ij}$ . Yet in practice, the training instances consist of  $(s, a, Q(s, a))$ -triplets, where  $y_i$  corresponds to the  $Q(s, a)$ -value while  $x_{ij}$  corresponds to the attribute values of the dimensions of the state  $s$  and action  $a$ .

Potts & Sammut (2005) found the regression tree induction algorithm applied here. In their research, Potts & Sammut (2005) introduce two similar methods to derive linear model trees. Both algorithms are suited to process a batch of training instances at once as well as to learn a model tree incrementally based on a continuous stream of instances. The tree induction is based on statistical tests performed to determine whether splitting a leaf node into two child nodes is significant, and to determine the position of the split. The first variant

of the induction technique, alias RD, analyses the difference in residual sum of squares. The second variant explores the distributions of positive and negative residuals and is called RA. The RD-approach requires calculating (in batch mode) and maintaining (in incremental mode) a linear model in each leaf and for each possible split, whereas only a single linear model is estimated in each leaf for the RA-technique. Consequently, the RA-algorithm is elaborated in the current research.

The model tree induction algorithm only provides binary splits, which are defined based on the distribution of the residuals (RA) of all observations assigned to the node into consideration. These residuals  $y_j - \hat{y}_j$  are calculated for each independent variable  $j$  and are equal to the difference between the actual (observed) value  $y_j$  and the value  $\hat{y}_j$  predicted by the linear model in the node. In the present study, for the sake of simplicity, the linear model equals the average of the dependent variable of all examples belonging to the node. According to these residuals, the algorithm divides the  $N$  instances of the node into consideration in two subsets: the  $N^+$  examples with non-negative residuals, denoted  $x_{ij}^+$ , are assigned to  $S^+$ ; and the  $N^-$  examples with negative residuals, labelled  $x_{ij}^-$ , belong to  $S^-$  (and  $N = N^+ + N^-$ ). For each dependent variable  $j$  following statistic is calculated (Potts & Sammut, 2005):

$$T_j^{(1)} = \frac{\bar{x}_j^+ - \bar{x}_j^-}{s_j \sqrt{\frac{1}{N^+} + \frac{1}{N^-}}}. \quad (2.11)$$

Here  $\bar{x}_j^+$  and  $\bar{x}_j^-$  equal the average of variable  $j$  in subset  $S^+$  and subset  $S^-$  respectively, and  $s_j$  denotes the pooled variance of this variable over both subsets.  $T_j^{(1)}$  tests for difference in means (Potts & Sammut, 2005).

Subsequently, the absolute differences  $z_{ij}^+ = |x_{ij} - \bar{x}_j^+|$  for all instances of  $S^+$  and  $z_{ij}^- = |x_{ij} - \bar{x}_j^-|$  for all instances of  $S^-$ , are determined. Based on these  $z$ -values  $T_j^{(2)}$  is defined, which tests for a difference in variances (Potts & Sammut, 2005):

$$T_j^{(2)} = \frac{\bar{z}_j^+ - \bar{z}_j^-}{w_j \sqrt{\frac{1}{N^+} + \frac{1}{N^-}}}. \quad (2.12)$$

Here  $\bar{z}_j^+$  and  $\bar{z}_j^-$  equal the average of the absolute differences of variable  $j$  in subset  $S^+$  and subset  $S^-$  respectively, and  $w_j$  refers to the pooled variance of  $z$ -values over both

subsets. Having calculated both  $T$ -values for all attributes, the split attribute is defined to be the variable corresponding to  $T = \max_{j,n} |T_j^{(n)}|$ . The corresponding attribute test value is determined as the average of the means  $\bar{x}_j^+$  and  $\bar{x}_j^-$  (Potts & Sammut, 2005).

Next, the probability  $\alpha$ , in which  $|t| > T$ , is retrieved from the Student's  $t$  distribution and compared to a predefined threshold  $\alpha_{split}$ , in order to check whether the proposed split is meaningful. Additionally, the growth of the tree is limited by estimating the contribution of the split to the overall tree accuracy by calculating (Potts & Sammut, 2005):

$$\delta = \frac{1}{s_{root}^2} \left( \frac{RSS}{N-d} - \frac{RSS_L + RSS_R}{N_L + N_R - d} \right). \quad (2.13)$$

Here  $L$  and  $R$  refer to the left ( $x_{ij} \leq test\ value$ ) and right ( $x_{ij} > test\ value$ ) child node respectively,  $RSS$  is the residual sum of squares,  $s_{root}^2$  is the standard deviance of the dependent variable calculated based on all cases and  $d$  is the number of dimensions. This value is also put against a user-defined threshold  $\delta_{split}$  to allow the algorithm to split the node, only if the contribution is large enough (Potts & Sammut, 2005). Besides this rule, the size of the tree can be influenced by only allowing the algorithm to split a node when this node contains a minimum number of instances.

Yet, in the current configuration of the algorithm, the regression tree is fitted off-line, which implies that the algorithm is not able to handle a continuous stream of data. Therefore, the regression tree induced by means of the batch regression tree algorithm has to be re-estimated from time to time. However, in the present research setting the number of instances rises rapidly, causing both the amount of memory required to store the training instances and the time required to estimate the regression tree to increase. Furthermore, as the agent faces a continuously changing environment and preferences, instances gathered a considerable time in the past may become outdated, obsolete or even invalid. Consequently, aiming at reducing memory requirements, speeding up the regression tree induction and mastering the accuracy of the tree model, a sliding window approach is considered, in which only a fixed number of the most recently encountered instances are used each time the tree is re-fitted.

The parameters, frequency of re-estimation  $f$  (or window shift) and the window size  $n$  (which represents the number of training instances used for each re-estimation), ought to be considered very carefully. The parameter  $f$  here indicates the number of training instances



gathered between two consecutive re-fits. First, re-fitting the regression tree more frequently (low  $f$ ) increases the ability to react to changes underlying the data stream, but increases the burden of updating the model tree. On the other hand, a low frequency of re-estimation (high  $f$ ) decreases both the total time required to keep the model up to date and the responsiveness of the model to changes in the data. Besides, using more data to train the model tree (high  $n$ ) increases the accuracy and enables the regression tree to level out temporary fluctuations. However, using a large number of training instances implies including instances gathered a long time ago, restricting the responsiveness of the algorithm to structural changes because the information recorded by these “old” instances may be outdated and in this way could interfere with the prediction of recent trends. This issue can be tackled by decreasing the importance of older instances by assigning weights to the instances, based on their age. This option is not further taken into account. Furthermore, as the number of instances used to estimate the tree increases, both the memory capacity required to store a large amount of instances and the computational time required to estimate a regression tree grow.

#### **Incremental Regression Tree Algorithm**

The drawbacks of the off-line batch induction algorithms require looking for an on-line tree induction algorithm as described in Potts & Sammut (2005). The main concepts of the incremental learning process are similar to the batch regression tree approach discussed in the previous paragraph. In essence, these algorithms differ in the fact that the incremental algorithm starts with an empty tree and updates the model tree on each encounter with a new instance. To this end, the incremental approach updates the statistics in each node of the tree that is affected by the training instance into consideration, instead of calculating these statistics based on a subset of the training set as is the case for the batch algorithm. If necessary, the incremental algorithm updates the existing tree by either creating an additional split or pruning a number of existing leaves (Potts & Sammut, 2005). The latter action allows the algorithm to correct previously defined splits which are no longer significant based on the most recently observed instances by comparing the  $\alpha$ -value of the node to a predefined threshold level  $\alpha_{prune}$ . After updating the model tree, the training instance can be discarded. As a result, it is not required to store the training instances recorded during

the learning process. Consequently, the incremental approach saves memory space.

In addition to this advantage, an intrinsic feature of the incremental algorithm consists of the ability of handling a continuous stream of data without having to re-train the model tree from scratch when a new instance becomes available. However, the gradual updating of the tree implies that the sequence in which the training instances are processed influences the structure of the regression tree to a large extent. This causes an incremental model tree to reveal a different structure from a batch regression tree, which is fitted on the same dataset but drawn up examining a large number of training instances at once.

#### **Batch/Incremental Regression Tree Algorithm**

In order to tackle the computational limitations of the batch algorithm and to reduce the sequence effects of the incremental algorithm, the current research proposes a hybrid batch/incremental model tree induction method combining both algorithms. To start with, the batch tree induction algorithm is applied to fit a model tree based on the first  $n$  instances collected. Thereafter, the resulting tree is updated on-line by applying the incremental induction algorithm when a new instance enters the system. The algorithm re-estimates the tree from scratch, based on the  $n$  most recent instances at a rate  $f$ . This approach allows the algorithm to formulate an initial tree, which already captures the most significant splits deduced from the data available for the batch algorithm. The existence of such structure facilitates the updating process within the incremental learning process. Because of this, it is assumed that the batch/incremental algorithm performs better than the incremental approach.

This hybrid technique is comparable to the batch algorithm, because it requires storing the  $n$  most recently gathered instances as well. Yet the incremental updating process weakens the influence of the frequency  $f$  of re-fitting the model tree on the accuracy of the resulting model, while decreasing the computational burden.

To reduce the computational burden even more, an additional rule is introduced, indicating whether or not the resulting model improves significantly. Consequently, the algorithm can decide whether these repeated batch re-fitting actions are required and, if not, that it can safely switch to incremental updating. However, this rule also indicates when the es-

timated accuracy of the model drops due to persistent changes in the data. In this case, the algorithm can decide to restart the recurring batch re-fitting. This rule is founded on a traditional statistical one-tailed  $t$ -test, which verifies whether the average of the SE (squared errors) of the  $n_{recent}$  recent training instances ( $\bar{\mu}_{recent}$ ) differs significantly from the average error of the preceding  $n_{previous}$  training instances ( $\bar{\mu}_{previous}$ ). Each SE-value is calculated when the current training instance occurs and before the tree model is updated according to this instance. For the stopping rule, the statistical hypotheses can be formulated as follows:

$$\begin{aligned} H_0 &: \mu_{recent} = \mu_{previous}, \\ H_1 &: \mu_{recent} < \mu_{previous}. \end{aligned} \tag{2.14}$$

The batch re-fitting continues as long as  $H_0$  is rejected, as this signifies that the average SE still decreases and there is still potential to improve the tree model. The hypotheses of the starting rule look as follows:

$$\begin{aligned} H_0 &: \mu_{recent} = \mu_{previous}, \\ H_1 &: \mu_{recent} > \mu_{previous}. \end{aligned} \tag{2.15}$$

The algorithm re-starts the batch re-fitting when  $H_0$  can be rejected. In this case, rejecting  $H_0$  denotes that the average of SE increases, and that the accuracy of the model decreases, probably due to changes in the data. The number of instances considered when calculating  $\bar{\mu}_{recent}$  and  $\bar{\mu}_{previous}$  (in this case  $n_{recent}$  and  $n_{previous}$  respectively), as well as the probability of accepting/rejecting these hypothesis (in this case  $\alpha_{batchstop}$  and  $\alpha_{batchstart}$ ), are passed on to the algorithm a priori.

As is the case for the incremental approach, the batch/incremental algorithm can discard the training instances after the batch training and the subsequent updating of the model tree. The major advantage of this on-line approach with regard to the batch algorithm consists of the decrease in the computational time required to update the tree compared to the time required to fit the model tree from scratch every time a new instance arrives. The next

section empirically evaluates the described batch, incremental and the batch/incremental induction methods.

### Case Study

Remember that the tree induction approach required in the current research has to be able to: (1) handle a numeric dependent variable, (2) be responsive to structural changes, (3) handle a continuous stream of data, and (4) handle a large amount of data.

The three model tree induction techniques proposed are particularly suited to deal with a numeric dependent variable. Furthermore, the previous paragraphs also showed that the nature of these algorithms enable processing a large and continuous stream of data. The second requirement refers to the term “concept drift”, which causes a model to become outdated as the data underlying the model tend to change. In this context, the following dimensions affect the responsiveness of the modelling algorithm: (1) the noise present in the training data, (2) the speed of the drift, (3) the extent of the drift, and (4) the presence of additional irrelevant attributes (Widmer & Kubat, 1996).

To assess the ability of the presented model tree algorithms to cope with these aspects, a case study is used, which consists of a synthetic problem using drifting hyperplanes as described in Kolter & Maloof (2005). Even though this synthetic problem is not directly related to the current research area, it is particularly suited to demonstrate the impact of these aspects on the responsiveness of the algorithms, because it enables the user to control for noise, the speed and extent of the drift and the presence of irrelevant attributes.

The training instances consist of ten numeric variables  $x_i$ , which are uniformly distributed in the range  $[0, 1]$ . The data include four different concepts with respect to the regressor  $y$ , which equals  $(x_j + x_{j+1} + x_{j+2})/3$ , where  $j$  is 1 for the first, 2 for the second, 4 for the third concept and 7 for the last concept. Each concept lasts for  $\tau$  time steps. Furthermore, a parameter  $\nu$  is introduced to account for the speed of the concept drift and denotes the number time steps required for the new concept to become effective.

During this transition period ( $t > \tau - \nu$ ) the regressor  $y$  corresponds to a weighted average of  $y$  according to the old concept  $C_i$  and the new one  $C_{i+1}$ :  $y = [(\tau - t)/\nu] y_{C_i} + [(t - (\tau - \nu))/\nu] y_{C_{i+1}}$ . Large values of  $\nu$  indicate a rather gradual drift, while a value of

$\nu$  close to zero signifies an abrupt concept drift or so-called concept shift (Widmer & Kubat, 1996), as displayed in figure 2.1. The presence of noise is introduced into the synthetic example by adding a random variate in the interval  $[-\epsilon, +\epsilon]$  to  $y$  (Kolter & Maloof, 2005). In each concept, all of the ten attributes, also the ones which are considered to be irrelevant to the prevailing concept, are included in the training instances. To measure the predictive power of the trees, a test set of 100 samples is generated. At each time step, for each of these test samples the actual value according to the prevailing concept is compared to the prediction deduced from the tree by means of the root mean squared error (RMSE). The accuracy of the prediction is expressed as the average of the RMSE-values of these 100 test instances.

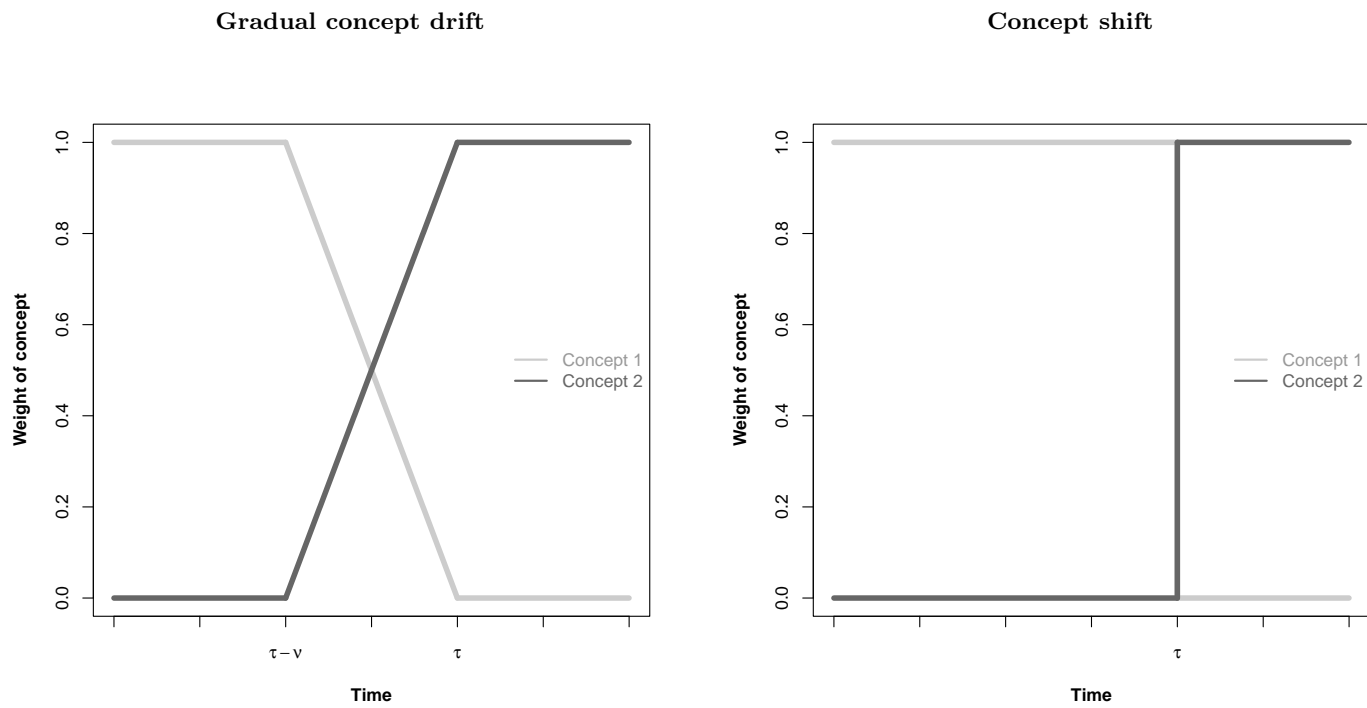


Figure 2.1: Weight of concept

The parameter  $\tau$  is fixed to 1,000. Four scenarios are examined based on the values of  $\nu$  and  $\epsilon$ . The value of the speed of the drift  $\nu$  is set to 500 to simulate a gradual drift and to 0 to simulate an abrupt concept shift. The parameter  $\epsilon$  equals 0.01 when studying the influence of noise, and 0.00 when disregarding noise. Furthermore, the impact of the rate of re-fitting the tree  $f$  and number of instances used to execute this re-estimation  $n$  on the predictive accuracy of the tree, is analysed by varying these parameters. An overview of the remaining parameter settings is summarized in table 2.4.

Parameter		Value(s)
<i>Standard model tree settings</i>		
Significance possible split	$\alpha_{split}$	0.00001
Significance actual split	$\alpha_{prune}$	0.1
Contribution possible split	$\delta_{split}$	0.0005
Minimum number of instances in node		5
<i>On-line model tree settings</i>		
Number of instances used for re-training	$n$	100/250/500
Re-training frequency	$f$	1/250/100/500
<i>Batch/Incremental tree settings</i>		
Significance starting rule	$\alpha_{batchstart}$	0.01
Significance stopping rule	$\alpha_{batchstop}$	0.05
Number of most recent instances	$n_{recent}$	100
Number of previous instances	$n_{previous}$	100
<i>Simulation settings</i>		
Duration of concept	$\tau$	1000
Speed of drift	$\nu$	0/500
Noise	$\epsilon$	0/0.01

Table 2.4: Parameter settings for model tree induction algorithms

The results are illustrated in figures 2.2 to 2.5 and tables 2.5 and 2.6 for the on-line batch induction algorithm, and figures 2.6 to 2.9 and tables 2.7 and 2.8 for the batch/incremental induction approach. The columns of these tables labelled “Time $x$ ” estimate the time required for the algorithm to converge to the best attainable result after the concept  $x$  is fully introduced, which is measured by the range of lowest RMSE. The average RMSE in the tables represents the average RMSE within this area of convergence.

In case of the batch algorithm, it is shown that it is necessary to re-train the model tree more frequently in order to maintain a high accuracy (i.e. low average RMSE). Concerning the number of training instances used to re-train, the analysis indicates that a low number

of training instances cannot guarantee a high predictive accuracy (e.g.  $n = 100$ ). Yet, a high number of training instances impedes the responsiveness of the model to changes underlying the data (e.g.  $n = 500$ ). With respect to the batch/incremental algorithm, the preceding findings are also valid, although one can easily infer from the graphs that the impact of these parameters is less pronounced compared to the case of the batch approach. As can be derived from the graphs, the speed of the concept drift does influence the shape of the average RMSE curve. However, it does not impact the validity of the conclusions. Finally, these figures reveal that the presence of noise does not affect the accuracy obtained by the algorithms.



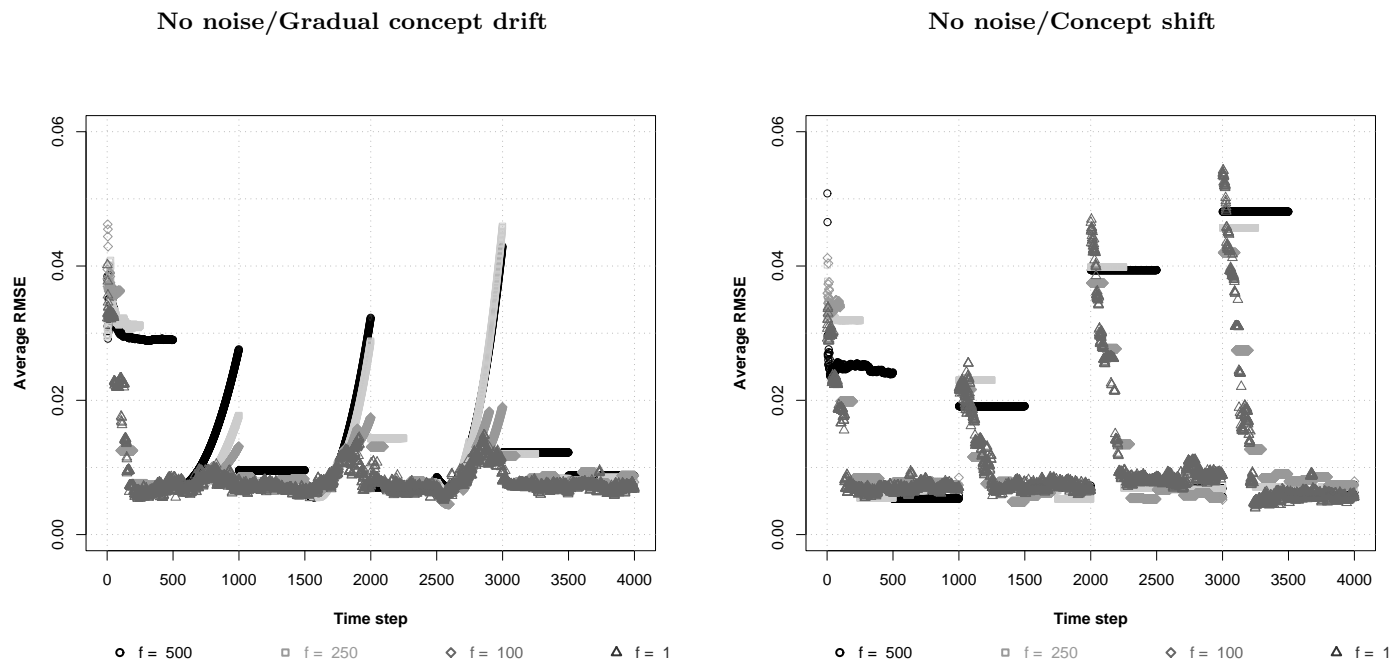


Figure 2.2: Impact of parameter  $f$  on accuracy of the batch induction algorithm ( $n = 250$ ) estimated on data **excluding noise**

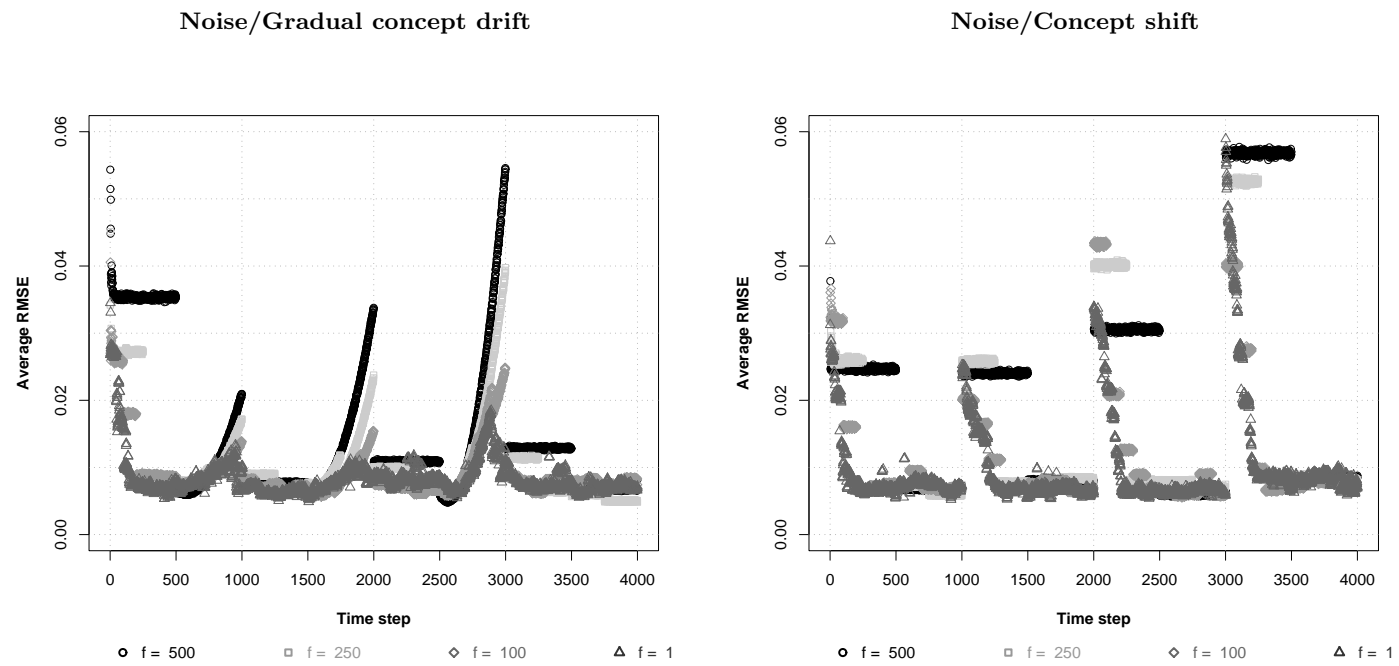


Figure 2.3: Impact of parameter  $f$  on accuracy of the batch induction algorithm ( $n = 250$ ) estimated on data **including noise**

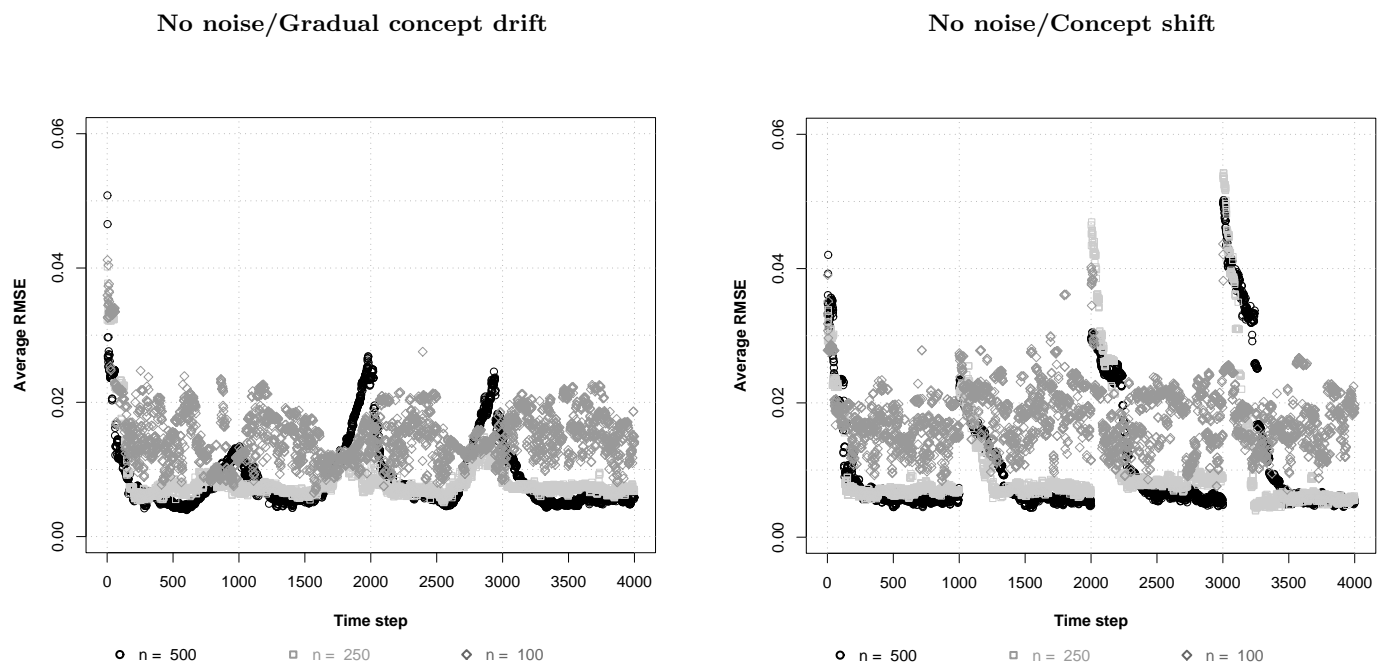


Figure 2.4: Impact of parameter  $n$  on accuracy of the batch induction algorithm ( $f = 1$ ) estimated on data **excluding noise**

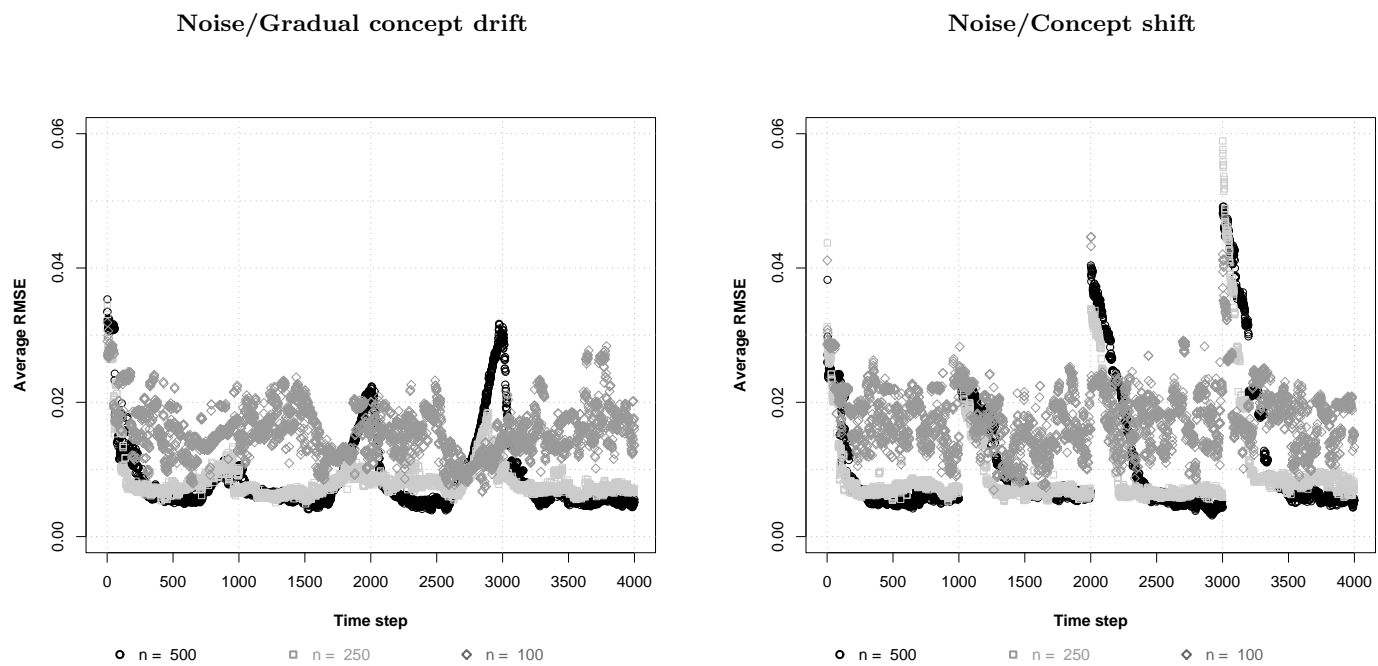


Figure 2.5: Impact of parameter  $n$  on accuracy of the batch induction algorithm ( $f = 1$ ) estimated on data **including noise**



Figure 2.6: Impact of parameter  $f$  on accuracy of the batch/incremental induction algorithm ( $n = 250$ ) estimated on data **excluding noise**

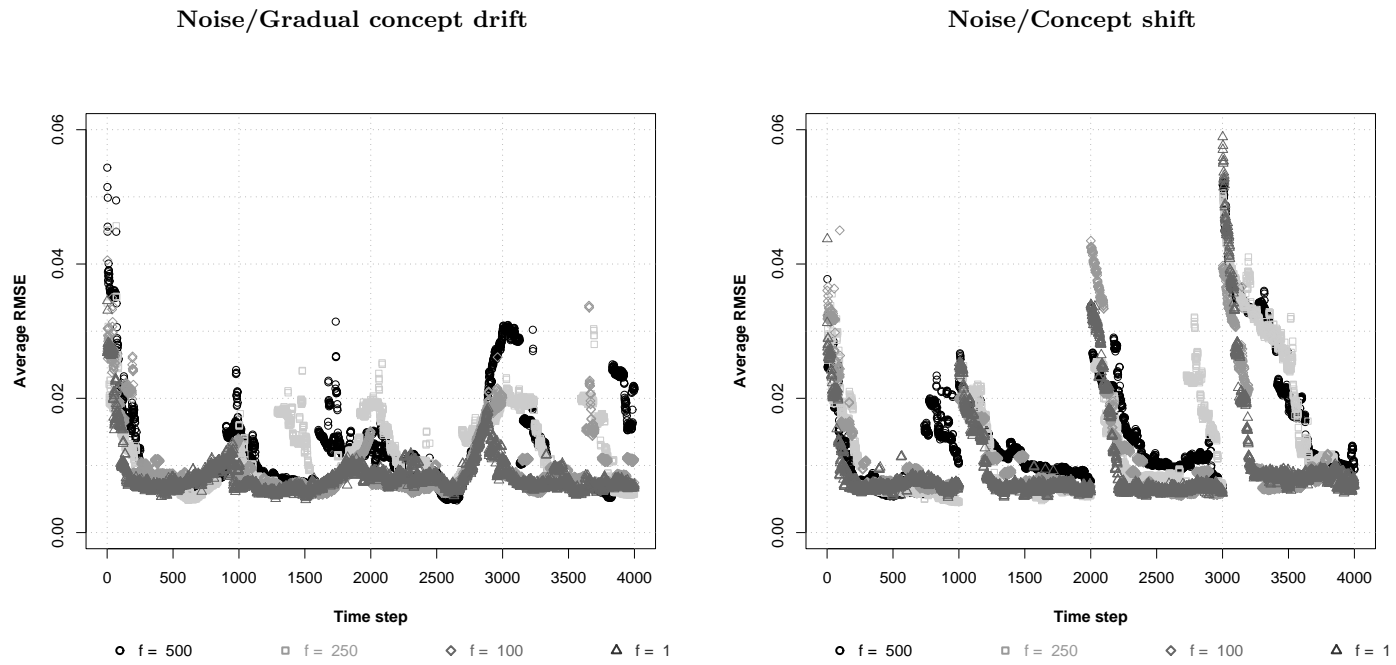


Figure 2.7: Impact of parameter  $f$  on accuracy of the batch/incremental induction algorithm ( $n = 250$ ) estimated on data **including noise**

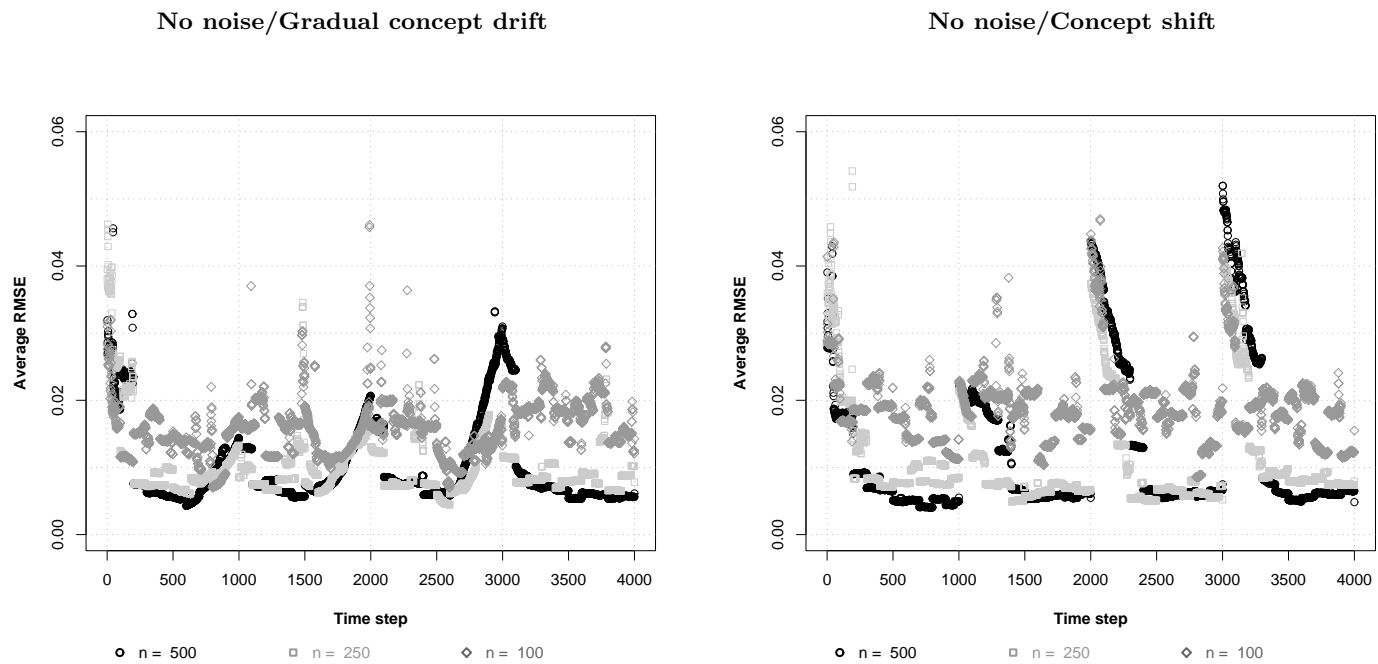


Figure 2.8: Impact of parameter  $n$  on accuracy of the batch/incremental induction algorithm ( $f = 100$ ) estimated on data **excluding noise**

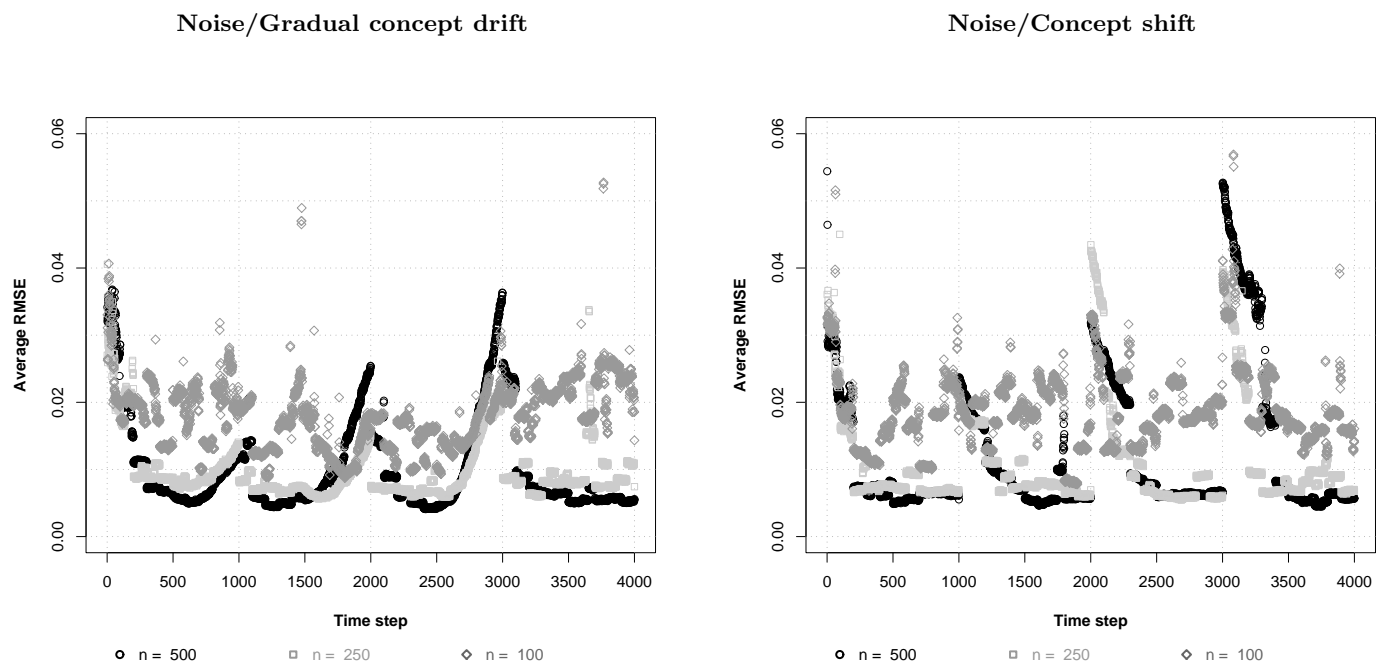


Figure 2.9: Impact of parameter  $n$  on accuracy of the batch/incremental induction algorithm ( $f = 100$ ) estimated on data **including noise**



$f$	Time1	Avg RMSE1	Time2	Avg RMSE2	Time3	Avg RMSE3	Time4	Avg RMSE4
<i>No noise/Gradual concept drift</i>								
500	500	0.00863	1500	0.00642	2001	0.01214	3500	0.00877
250	250	0.00774	1001	0.00697	2250	0.00756	3500	0.00797
100	200	0.00770	1100	0.00742	2100	0.00699	3400	0.00786
1	148	0.00698	1001	0.00693	2111	0.00674	3234	0.00712
<i>No noise/Concept shift</i>								
500	500	0.00535	1500	0.00699	2500	0.00687	3500	0.00628
250	250	0.00645	1250	0.00635	2250	0.00698	3250	0.00691
100	200	0.00781	1400	0.00634	2300	0.00598	3200	0.00870
1	132	0.00674	1237	0.00721	2204	0.00854	3209	0.00581
<i>Noise/Gradual concept drift</i>								
500	500	0.00784	1001	0.00773	2001	0.00955	3500	0.00674
250	250	0.00833	1250	0.00701	2001	0.00853	3262	0.00580
100	200	0.00858	1100	0.00685	2001	0.00684	3200	0.00629
1	143	0.00801	1011	0.00643	2053	0.00797	3134	0.00720
<i>Noise/Concept shift</i>								
500	500	0.00677	1500	0.00804	2500	0.00589	3500	0.00850
250	250	0.00658	1250	0.00789	2250	0.00766	3250	0.00846
100	200	0.00735	1300	0.00719	2300	0.00696	3200	0.00759
1	94	0.00733	1207	0.00680	2185	0.00662	3197	0.00836

Table 2.5: Impact of parameter  $f$  on accuracy of the batch induction algorithm ( $n = 250$ )

$n$	Time1	Avg RMSE1	Time2	Avg RMSE2	Time3	Avg RMSE3	Time4	Avg RMSE4
<i>No noise/Gradual concept drift</i>								
500	161	0.00563	1119	0.00603	2115	0.00662	3242	0.00505
250	148	0.00698	1001	0.00693	2111	0.00674	3234	0.00712
100	140	0.01410	1031	0.01454	2022	0.01452	3001	0.01627
<i>No noise/Concept shift</i>								
500	130	0.00630	1337	0.00584	2307	0.00653	3332	0.00618
250	132	0.00674	1237	0.00721	2204	0.00854	3209	0.00581
100	177	0.01577	1091	0.01881	2083	0.01733	3167	0.01601
<i>Noise/Gradual concept drift</i>								
500	239	0.00624	1044	0.00588	2079	0.00574	3225	0.00562
250	143	0.00801	1011	0.00643	2053	0.00797	3134	0.00720
100	187	0.01491	1011	0.01581	2017	0.01398	3001	0.01655
<i>Noise/Concept shift</i>								
500	178	0.00573	1331	0.00681	2342	0.00559	3316	0.00666
250	94	0.00733	1207	0.00680	2185	0.00662	3197	0.00836
100	186	0.01737	1162	0.01450	2134	0.01731	3087	0.01754

Table 2.6: Impact of parameter  $n$  on accuracy of the batch induction algorithm ( $f = 1$ )

$f$	Time1	Avg RMSE1	Time2	Avg RMSE2	Time3	Avg RMSE3	Time4	Avg RMSE4
<i>No noise/Gradual concept drift</i>								
500	277	0.00772	1229	0.00704	2036	0.00800	3444	0.00738
250	266	0.00797	1423	0.00661	2054	0.01066	3160	0.00729
100	200	0.00805	1100	0.00879	2100	0.00787	3100	0.00877
1	148	0.00698	1001	0.00693	2111	0.00674	3234	0.00712
<i>No noise/Concept shift</i>								
500	373	0.00681	1238	0.00632	2268	0.00807	3622	0.01052
250	227	0.00707	1342	0.00831	2433	0.00909	3406	0.00726
100	200	0.00891	1202	0.00651	2273	0.00629	3277	0.00834
1	132	0.00674	1237	0.00721	2204	0.00854	3209	0.00581
<i>Noise/Gradual concept drift</i>								
500	247	0.00733	1138	0.00781	2500	0.00582	3130	0.00928
250	223	0.00665	1047	0.01056	2233	0.00844	3378	0.01064
100	200	0.00876	1100	0.00679	2001	0.00700	3100	0.00784
1	143	0.00801	1011	0.00643	2053	0.00797	3134	0.00720
<i>Noise/Concept shift</i>								
500	183	0.00664	1485	0.00925	2383	0.01019	3629	0.01067
250	250	0.00592	1269	0.00634	2229	0.00830	3579	0.00992
100	200	0.00745	1300	0.00762	2300	0.00708	3200	0.00798
1	94	0.00733	1207	0.00680	2185	0.00662	3197	0.00836

Table 2.7: Impact of parameter  $f$  on accuracy of the batch/incremental induction algorithm ( $n = 250$ )

$n$	Time1	Avg RMSE1	Time2	Avg RMSE2	Time3	Avg RMSE3	Time4	Avg RMSE4
<i>No noise/Gradual concept drift</i>								
500	200	0.00626	1100	0.00689	2100	0.00719	3200	0.00618
250	200	0.00805	1100	0.00879	2100	0.00787	3100	0.00877
100	169	0.01334	1200	0.01394	2532	0.00978	3092	0.01824
<i>No noise/Concept shift</i>								
500	300	0.00545	1400	0.00588	2400	0.00609	3300	0.00619
250	200	0.00891	1202	0.00651	2273	0.00629	3277	0.00834
100	186	0.01714	1390	0.01684	2185	0.01766	3570	0.01596
<i>Noise/Gradual concept drift</i>								
500	300	0.00652	1100	0.00554	2200	0.00542	3200	0.00600
250	200	0.00876	1100	0.00679	2001	0.00700	3100	0.00784
100	400	0.01672	1170	0.01433	2128	0.01407	3091	0.02141
<i>Noise/Concept shift</i>								
500	200	0.00638	1400	0.00583	2300	0.00687	3400	0.00601
250	200	0.00745	1300	0.00762	2300	0.00708	3200	0.00798
100	200	0.01462	1800	0.00825	2100	0.01650	3200	0.01741

Table 2.8: Impact of parameter  $n$  on accuracy of the batch/incremental induction algorithm ( $f = 100$ )

Figures 2.10 and 2.11 and table 2.9 join the results of the batch ( $n = 250$  and  $f = 1$ ), the batch/ incremental ( $n = 250$  and  $f = 100$ ) and incremental methods, denoted B, B/I and I respectively. The graph also includes the predictive performance of a baseline model (denoted base), of which the prediction at a certain time step equals the average value of the regressor variable of the  $n$  preceding training instances. In order to enable a fair comparison,  $n$  equals 250 as well. The graph illustrates well that the accuracy achieved by the batch/incremental tree induction algorithm approaches the accuracy of the batch tree induction algorithm very closely. However, to reach this level of predictive performance, the latter technique is far more demanding regarding the computational requirements than the former, as the batch model tree has to be re-fitted every time while the batch/incremental model tree is only re-estimated from scratch at most every 100 instances and updated in-between. Furthermore, the graph shows that both approaches outperform the baseline model. Finally, figures 2.10 and 2.11 and table 2.9 also prove that the purely incremental approach performs well when no changes in the data occur. Yet, when concept drift does take place, the resulting tree is not able to adapt accordingly, causing the accuracy to drop considerably, and even causing the tree to perform worse than the baseline model.

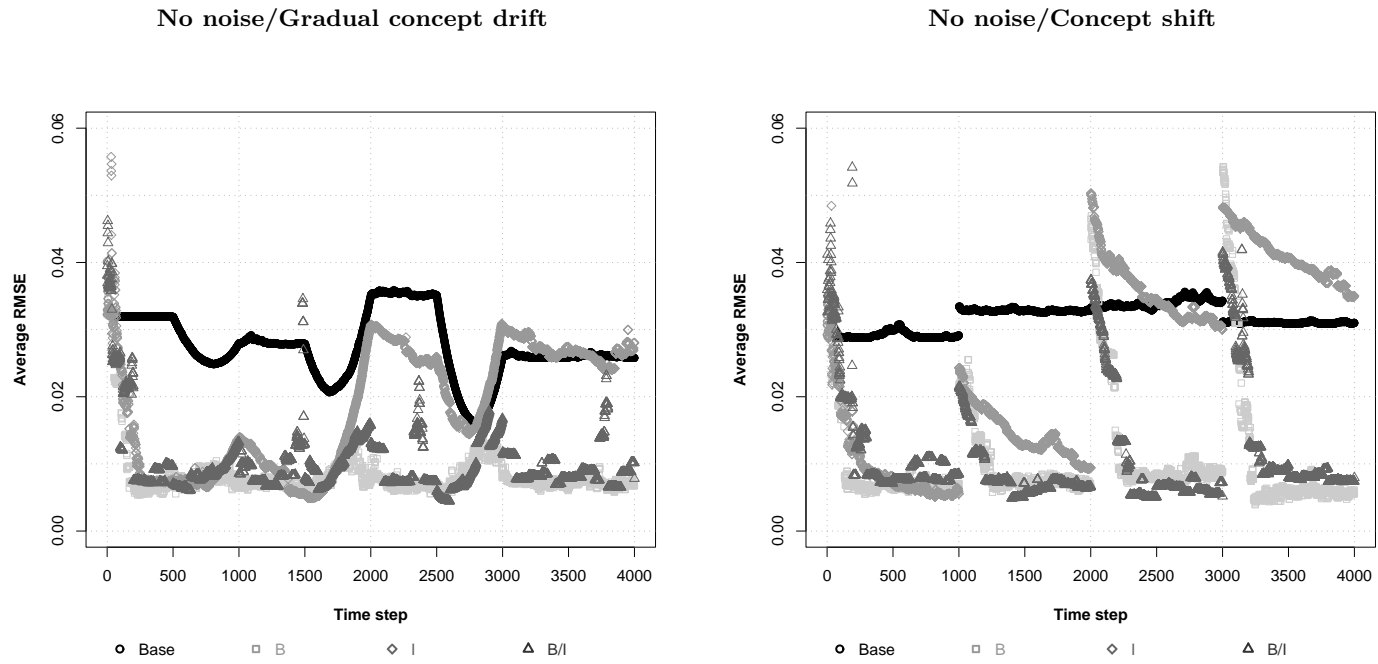


Figure 2.10: Comparison of tree induction algorithms ( $n = 250$  and  $f = 1$  for the on-line batch algorithm and  $f = 100$  for the batch/incremental algorithm) estimated based on data **excluding noise**

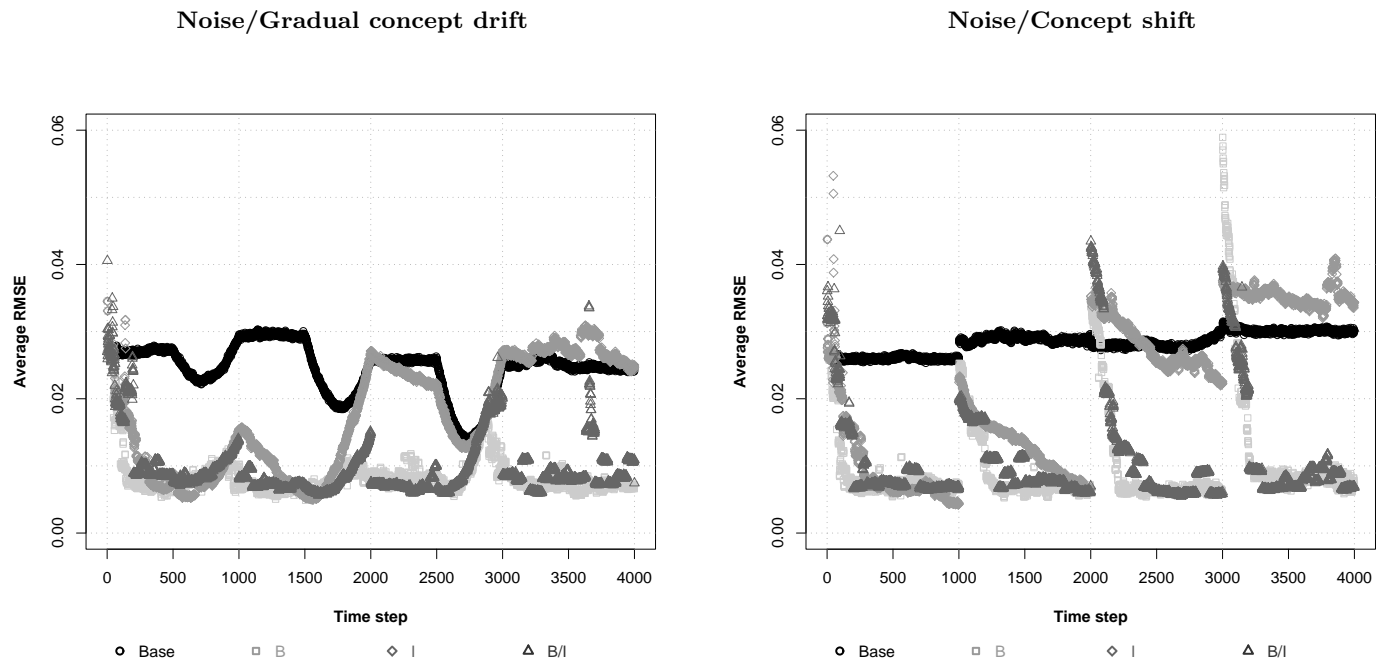


Figure 2.11: Comparison of tree induction algorithms ( $n = 250$  and  $f = 1$  for the on-line batch algorithm and  $f = 100$  for the batch/incremental algorithm) estimated based on data **including noise**

Alg.	Time1	Avg RMSE1	Time2	Avg RMSE2	Time3	Avg RMSE3	Time4	Avg RMSE4
<i>No noise/Gradual concept drift</i>								
BL	658	0.02550	1593	0.02162	2649	0.01776	3200	0.02597
B	148	0.00698	1001	0.00693	2111	0.00674	3234	0.00712
B/I	200	0.00805	1100	0.00879	2100	0.00787	3100	0.00877
I	222	0.00780	1356	0.00601	2601	0.01605	3700	0.02457
<i>No noise/Concept shift</i>								
BL	9	0.02903	1162	0.03282	2001	0.03348	3114	0.03104
B	132	0.00674	1237	0.00721	2204	0.00854	3209	0.00581
B/I	200	0.00891	1202	0.00651	2273	0.00629	3277	0.00834
I	302	0.00668	1796	0.01035	2564	0.03179	3706	0.03703
<i>Noise/Gradual concept drift</i>								
BL	603	0.02316	642	0.01962	618	0.01504	42	0.02502
B	143	0.00801	1011	0.00643	2053	0.00797	3134	0.00720
B/I	200	0.00876	1100	0.00679	2001	0.00700	3100	0.00784
I	319	0.00698	1307	0.00640	2593	0.01387	3107	0.02717
<i>Noise/Concept shift</i>								
BL	10	0.02603	1001	0.02890	2007	0.02791	3059	0.03009
B	94	0.00733	1207	0.00680	2185	0.00662	3197	0.00836
B/I	200	0.00745	1300	0.00762	2300	0.00708	3200	0.00798
I	278	0.00644	1664	0.00826	2495	0.02487	3465	0.03411

Table 2.9: Comparison of tree induction algorithms ( $n = 250$  and  $f = 1$  for the on-line batch algorithm and  $f = 100$  for the batch/incremental algorithm)



It can thus be concluded that the batch/incremental algorithm performs well - even in the presence of noise or after the introduction of a new concept -, while not being computationally or memory-wise as demanding as the batch algorithm. As a result, the batch/incremental technique is incorporated as the function approximator in the reinforcement learning algorithm.

## 2.4 Multi-Actor Reinforcement Learning

In some reinforcement learning problems, an agent can be contemplated as a multi-actor system, in which each action of this agent can in fact be determined by a number of sub-actions (Zennir & Couturier, 2005). For instance in the current research area, an agent corresponds to a household member who decides which activity he wants to execute at which location, given a certain state of the environment. Assume that such agent can select from four activities (e.g. in-home activities, out-of-home work, maintenance or leisure activities) and from four locations (e.g. home, location 1, 2 or 3). Instead of considering a joint decision fixing the activity and the location simultaneously, this decision can be divided into its two compounding sub-decisions, determining the activity and the location successively. In the former case of one global action, the agent faces sixteen feasible actions ( $= 4 \text{ activities} \times 4 \text{ locations}$ ), while in the latter case, the action set in each component includes only four feasible actions.

Clearly, the curse of dimensionality also leaves its mark here. After all, in the traditional reinforcement learning system the number of feasible actions increases rapidly with the number of alternatives for each sub-decision, which has repercussions on the computational and memory requirements of the learning process because all of these alternatives have to be explored. However in the multi-actor reinforcement setting, an increase in the number of alternatives for one sub-decision does not impact the size of the action sets of the remaining sub-decisions. Consequently, if a reinforcement learning problem, that is suitable for a multi-actor setting, can be reformulated so that the global reward equals the sum of the reward of these sub-actions, it is advantageous to design a multi-actor reinforcement learning system, as is substantiated in the remainder of this section.

In a multi-actor reinforcement learning system, an agent is composed of a number of

components - or also called modules or actors -, corresponding to a particular subaction  $a^i$  of the composite action  $a$  and incorporating an autonomous reinforcement learning system. This idea is visualized in figure 2.12 for the example described above. The learning process of a multi-actor agent is similar to that of a traditional reinforcement learning agent. First, the multi-actor agent observes the state of the environment. Next, the agent calls each of the action modules successively to determine the sub-actions, which jointly define the global action. This global action is communicated to the environment and executed. Subsequently, the agent receives a global reward from the environment, which serves as input to update the  $Q(s, a)$ -values within each of the modules.

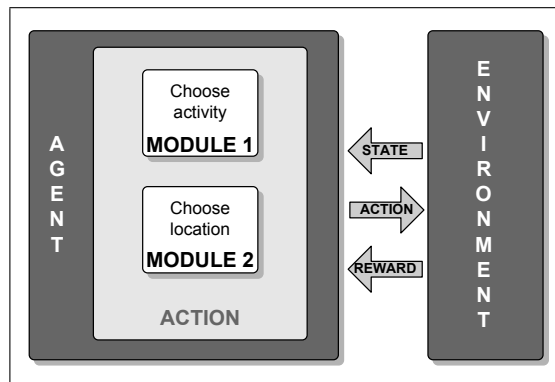


Figure 2.12: Multi-actor reinforcement learning system

Additionally, interactions between the components in the multi-actor reinforcement learning system may exist, which influence the outcome of the joint action (Zennir & Couturier, 2005). This is also the case in the current research area, in which the components founding activity-travel behaviour are supposed to be interrelated, as pointed out by Gärling *et al.* (1997) and Joh *et al.* (2002). To illustrate this for the example outlined above, assume a rather simple, tabular reward function as shown in table 2.10, and an additive effect of the rewards received for each of the sub-actions.

Composite action id	Activity	Location	Reward activity	Additional reward location
1	In-home activities	Home	3	5
2		Location 1		0
3		Location 2		0
4		Location 3		0
5	Out-of-home work	Home	4	0
6		Location 1		5
7		Location 2		4
8		Location 3		1
9	Out-of-home maintenance	Home	1	0
10		Location 1		6
11		Location 2		0
12		Location 3		4
13	Out-of-home leisure	Home	2	0
14		Location 1		1
15		Location 2		1
16		Location 3		8

Table 2.10: Rewards assigned to choice of activity and location

The optimal action disregarding interaction between the modules is composed of the sub-actions which maximise the local reward of each module individually. Taking a close look at the reward table, one can easily infer that the global optimal action disregarding interaction thus consists of working out-of-home - generating a reward of 4 for the activity module - at location 1 - yielding a reward of 5 for the location module. The total reward of this composite action (labelled composite action 6) is equal to 9 (=4+5). Yet, if the additive effect of location is accounted for when deciding on the activity to perform, the optimal joint action includes an out-of-home leisure activity - creating a reward of 2 for the activity module - at location 3 - giving a reward of 8 for the location module. In this case, the total reward of the system equals 10 (=2+8), which exceeds the optimal solution without considering interaction. This global optimal action is referred to as composite action 16.

In order to take these interactions into account and to enable the system to reach the global optimal solution, concepts from distributed reinforcement learning are borrowed (Littman & Boyan, 1993). In particular, experiences on individual rewards given the actions of preceding modules are shared between the modules. The learning scheme in figure 2.13 illustrates the difference between traditional reinforcement learning and the proposed multi-actor reinforcement learning (denoted traditional RIL and multi-actor RIL respectively).

When selecting a sub-action  $a^{(i)}$ , module  $M^{(i)}$  communicates this decision to module  $M^{(i+1)}$  and queries the latter on the best value he expects to benefit given the decision of module  $M^{(i)}$ , i.e.  $\hat{Q}_t^{(i+1)}(s, a^{(i+1)} | a^{(i)})$ . This way, module  $M^{(i)}$  incorporates the effect of selecting action  $a^{(i)}$  on the action values of the subsequent modules when updating its  $\hat{Q}^{(i)}$ -values (cf. methodology described in Littman & Boyan (1993)).

For this purpose, the  $\hat{Q}^{(i)}$ -value of module  $M^{(i)}$  is updated according to the following rule, as inspired on Littman & Boyan (1993):

$$\begin{aligned} \hat{Q}_{t+1}^{(i)}(s, a^{(i)}) \leftarrow & [1 - \alpha_{t+1}(s, a)] \hat{Q}_t^{(i)}(s, a^{(i)}) \\ & + \alpha_{t+1}(s, a) \left\{ r^{(i)}(s, a^{(i)}) + \max_{a^{(i+1)}} \left[ \hat{Q}_t^{(i+1)}(s, a^{(i+1)} | a^{(i)}) \right] \right\}. \end{aligned} \quad (2.16)$$

This formula differs from the updating rule 2.8 of the original  $Q$ -learning in the value of

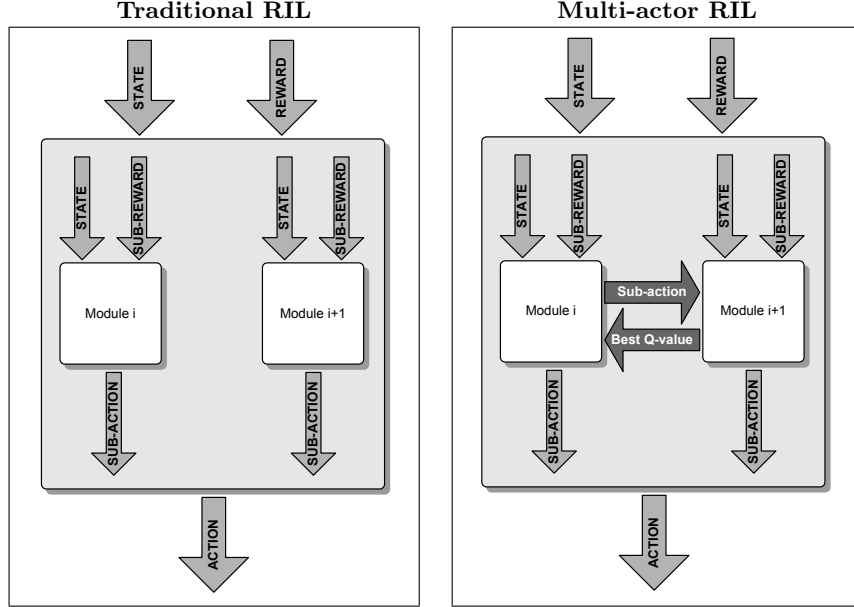


Figure 2.13: Learning schemes

the “new” estimate  $r^{(i)}(s, a^{(i)}) + \max_{a^{(i+1)}} [\hat{Q}_t^{(i+1)}(s, a^{(i+1)} | a^{(i)})]$ , which is now composed of the immediate reward  $r^{(i)}(s, a^{(i)})$  of action  $a^{(i)}$  in state  $s$  and the estimate of the best value that module  $M^{(i+1)}$  can attain given state  $s$  and action  $a^{(i)}$ , i.e.  $\hat{Q}_t^{*(i+1)}(s, a^{(i+1)} | a^{(i)})$ . Table 2.11 outlines the learning process in the proposed multi-actor reinforcement learning scheme.

---

Initialize each entry $\hat{Q}^{(i)}(s, a)$ in the $Q^{(i)}$ -table of module $M^{(i)}$ .
Repeat:
Observe state $s$ .
For each module $M^{(i)}$ :
Select sub-action $a^{(i)}$ .
Execute composite action $a$ .
Observe next state $s'$ .
For each module $M^{(i)}$ :
Observe sub-reward $r^{(i)}(s, a^{(i)})$ .
Query $\hat{Q}_t^{(i+1)}$ -value of module $M^{(i+1)}$ given action $a^{(i)}$ .
Calculate $\hat{Q}_{t+1}^{(i)}(s, a^{(i)})$ based on $(s, a^{(i)}, r(s, a^{(i)}))$ -triplet according to Equation 2.16.
Update $Q^{(i)}$ -table.

---

Table 2.11: Multi-actor reinforcement learning process

To assess the applicability of this multi-actor reinforcement learning approach in the

present study area, the example sketched above is implemented. The label of the optimal composite action as well as the evolution of the total value of these actions generated in the course of 100 learning episodes by the traditional modular and multi-actor reinforcement learning approach (labelled traditional RIL and multi-actor RIL respectively), are presented in figure 2.14. This figure reveals that both algorithms are able to learn a global optimal action after an initial learning phase, during which the optimal composite action is selected randomly. However, the optimal solution generated by the multi-actor reinforcement learning approach converges to the global optimal action (action value of 10), while the traditional modular reinforcement learning algorithm converges to the suboptimal action (action value of 9). The example presented here clearly demonstrates that the proposed multi-actor reinforcement learning approach enables the system to reach a higher global solution due to the incorporation of interactions between the decision units.

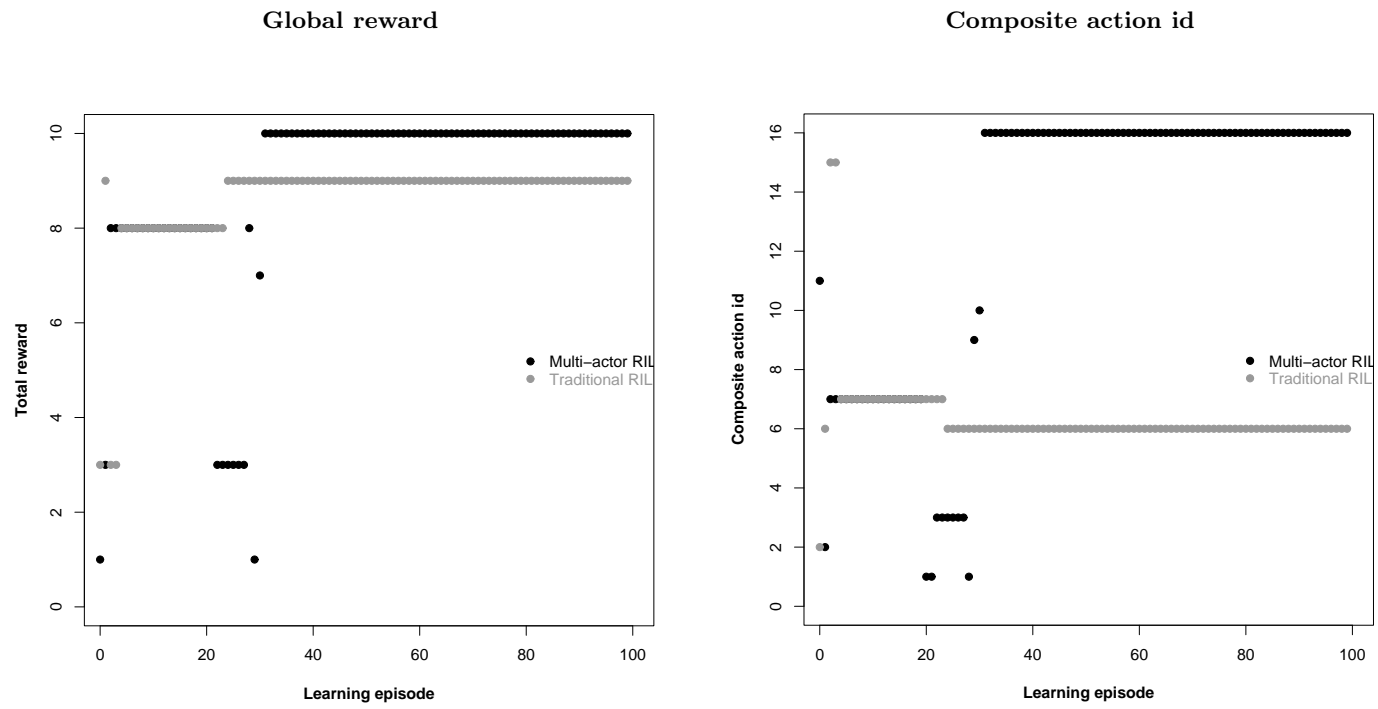


Figure 2.14: Evolution of optimal composite action

Obviously, the order in which the decisions are selected does not influence the ability of attaining the global optimal solution. Furthermore, due to multi-actor reinforcement learning the optimal composite action can also be learnt in a system containing more than two decision components.

## 2.5 Related Research Efforts Applying Reinforcement Learning

This section provides a brief summary of research applying reinforcement learning in the field of activity-based travel-demand modelling. In their attempt to incorporate learning and adaptation processes in activity-travel behaviour modelling, Arentze & Timmermans (2003) propose a framework to simulate activity-travel choices, in which the individual agent explores his environment and learns from past experiences. To meet the curse of dimensionality of the state and action spaces, the authors introduce two changes to the traditional reinforcement learning system. First they also assume a modular framework in which decisions on subactions are made sequentially instead of simultaneously. Consequently, the decisions of the subactions preceding the considered subaction are supposed to be given and can thus be treated as part of the state of this lower-level decision component.

Furthermore, Arentze & Timmermans (2003) implement an incremental CHAID-based tree mechanism to map state dimensions to state conditions, which maximize the homogeneity within these condition groups with respect to the reward values. These state conditions are then used as feasible state values within the reinforcement learning approach. Each time new state conditions are created or existing state conditions are joined - i.e. when splitting or pruning nodes - the  $\hat{Q}$ -values are estimated non-incrementally, based on all  $\hat{Q}(s, a)$ -values within the relevant subset. Even though the experiments are conducted on a hypothetical choice situation which covers route and destination choice, and choice of transport mode, the results show that this system is able to handle the full complexity of real-world situations. Additionally, this framework enables to take interactions between choice facets into account. Finally, the research also lifted the veil on the ability of the system to deal with a dynamic environment.

Charypar & Nagel (2005) apply reinforcement learning to generate daily activity plans. To accomplish this goal, Charypar & Nagel (2005) divide a day into a number time slots.



At every time step, an agent decides whether he wants to extend the duration of the current activity or whether he wants to execute another activity. Additionally, the authors assume a fixed sequence in which the activities occur. The action space is thus limited to either stay (i.e. prolong the duration of the current activity with one time unit) or move (i.e. execute the next activity of the sequence). The state space is composed of the activity, the starting time of the activity, and the current activity duration.

Although the framework is tested on a rather small-scale example, the analyses conducted by Charypar & Nagel (2005), prove that  $Q$ -learning is particularly suited to include within-day re-planning caused by unexpected events which interfere with the execution of the “optimal” activity-travel sequence (e.g. due to traffic congestion), even without explicitly modelling rescheduling. After all, during the learning phase the agent learns to behave optimally, even if he is faced with states which he would never have reached when executing the “optimal” plan, thanks to exploration of the state space. In other words,  $Q$ -learning enables selecting the best continuation of the plan in every feasible state.

Janssens *et al.* (2005) extend the time allocation approach introduced by Charypar & Nagel (2005) by incorporating location choice based on travel time. To this end, travel mode information is recorded in the fixed sequence as well. These authors also conclude that the  $Q$ -learning technique enables the agent to respond gracefully to unforeseen incidents.

## 2.6 Conclusions

This chapter supplies an overview of the concepts defining reinforcement learning in general and  $Q$ -learning in particular. This technique provides a solid ground of the modelling framework designed in the present research as it simulates the human way of learning through trial-and-error interactions with a dynamic environment while not requiring an explicit model of this environment. Yet, reinforcement learning does entail a number of drawbacks, and, as a result, restricting its applicability in research areas containing large state-action spaces.

To meet these limitations, the current research proposes to extend the  $Q$ -learning framework with a regression tree function approximator to enable generalization of the state-action spaces. The regression tree induction algorithm founding this function approximation is introduced by Potts & Sammut (2005) and is discussed briefly in the current chapter. Although

this regression tree induction algorithm can either be used as a batch or as an incremental induction algorithm, both variants include some restrictions. Therefore, the present effort suggests a hybrid batch/incremental version aiming at countering these constraints. A case study illustrates the added value of this hybrid batch/incremental regression tree induction algorithm.

Additionally, this chapter examines the ability to incorporate interactions between decision components in the reinforcement learning framework. To this end, multi-actor reinforcement learning is introduced. The functioning and applicability of this technique is demonstrated by means of an example originating from the area of activity-based modelling.

Finally, this chapter reviews related research efforts including reinforcement learning to model activity-travel behaviour.

## Chapter 3

# Data Pre-processing

### 3.1 Introduction

Activity-based travel-demand models are generally calibrated based on data observed from a number of individuals, who are assumed to be representative of the (synthetic) population the model intends to simulate. As these data originate from individuals, who differ in their preferences and wants, and who face different opportunities and constraints, the data are likely to contain very divergent activity-travel sequences with respect to the activity types included in the individual schedule, the order of the activity episodes, the duration of these activity episodes, the geographical dispersion of the locations visited, the travel modes used to reach these locations, etc. Consequently, it is generally accepted that sequences has to be analysed both in space and time (Joh *et al.*, 2001; 2002; 2007; Kulkarni & McNally, 2001; Kwan, 2000; Pas, 1983; Schlich, 2001; Wilson, 1998a;b; 2001; 2008). This way, it is possible to identify groups of individuals revealing similar activity-travel patterns and to attach socio-demographic profiles to each of these groups (Wilson, 1998b).

This chapter first gives a brief description of the available data set. Next, section 3.3.1 introduces the reader to a widely applied technique, denoted sequence alignment method, which aims at comparing activity patterns while taking the sequential aspect into consideration (Joh *et al.*, 2007; Wilson, 1998a;b; 2001). Yet only a limited number of research efforts do account for multidimensional sequences including spatial characteristics. Therefore, section 3.3.2 provides an overview of an extension, described in Wilson (2008), to the

sequence alignment method, which enables comparing the observed sequences both in space and time. However, the presented technique entails one major drawback: it is mainly suited to compare sequences in a small study area. To this end, section 3.3.3 attempts to design a spatio-temporal dissimilarity measure which enables determining similarities between geographically dispersed sequences, regardless of their absolute geographical location. Next, after having analysed the differences in the individual activity-travel sequences based on the activity type, duration and location, section 3.4.1 aims at identifying groups of individuals displaying similar activity-travel behaviour. Finally, the recorded socio-demographic attributes are linked to these groups in section 3.4.3 in order to formulate a number of socio-demographic profiles which can be applied to divide the synthetic population into a number of subsets matching the groups discerned in the preceding section.

## 3.2 Data

### 3.2.1 Data Collection Effort

The data used in this research stem from a project entitled "An Activity-Based Approach for Surveying and Modelling Travel Behaviour", aiming at collecting seven-day activity-travel diaries recorded from one adult member of 2,500 households across Flanders (Belgium) (Bellemans *et al.*, 2008; Janssens & Wets, 2005). For each day of the week, an activity-travel diary records a sequence - sometimes also called an activity schedule - containing all activities executed on that day along with their location, starting time and duration, travel mode used to get to the location, travel time and travel party. In the current study, each of these activities along with their characteristics is referred to as an activity episode. Furthermore, the terms "sequence" and "patterns" both refer to this ordered assembly of activity episodes, belonging to one day, and are used interchangeably. The activities used in this research are classified into thirteen activity categories: grocery shopping (G), non-daily shopping (N), education (D), social activities (O), leisure (L), bring/get activities (B), touring (T), working (W), services (V), out-of-home eating (E), sleeping (S), in-home activities (H) and a remainder category (R).

Beside these activity-travel characteristics, the respondents are asked to fill out an indi-

vidual and household survey covering socio-demographic variables, such as age, household composition, income, educational background and work situation. Furthermore, this research analyses the data recorded by means of a GPS-based personal digital assistant. As a consequence, the collected diaries contain geographical information, of which the importance emerges in the next section. Nevertheless, the data used in this research embrace only a part of the gathered individual activity-travel diaries within the project, as the entire dataset is not available at the time of writing this manuscript due to backlogs in inputting and cleaning the gathered data. Additionally, some of the diaries are not included as the data goes through an extensive data cleaning process in order to assure sufficient quality of the data constituting the basis of the algorithm described in chapter 4.

### 3.2.2 Some Descriptive Statistics

The data consist of 594 activity-travel patterns of 280 distinct individuals. Table 3.1 contains some information on the composition of these sequences. The column titled “%seq” quantifies the percentage of sequences in which the activity is observed at least once. The total duration per sequence and the average duration per episode are based only on the sequences which report at least one episode of the activity type into consideration. The observed sequences include minimum 3, maximum 17 and on average 4.82 activity episodes. Each of these patterns contains one sleeping episode preceding the activity sequence and one at the end of the sequence. In addition, the majority of the sequences (98%) records at least one in-home activity episode. Together with sleeping, the in-home activities are the only activity type executed on average more than once per sequence; the average total duration spent on these two activity categories is by far the highest. Though a note has to be made on the total duration of the sleeping activity: this total duration includes both the morning sleep episode and the evening sleep episode and thus actually spans two days, while the total duration of the non-sleeping activities occur within only one diary day. Both the time spent on each in-home activity episode and the total duration spent on this activity in the course of one sequence display a large variation, as is reflected by the standard deviance. 33% patterns comprise one or more working episode(s). When executed, the working activity is ranked third when examining the average of the total duration per sequence as well as of the average

duration per episode. Touring or the remainder activity categories are observed in none of the sequences.

Activity type	%seq	# episodes per seq				Total duration per seq				Duration per episode			
		<i>Min</i>	<i>Max</i>	<i>Avg.</i>	<i>Sd</i>	<i>Min</i>	<i>Max</i>	<i>Avg</i>	<i>Sd</i>	<i>Min</i>	<i>Max</i>	<i>Avg</i>	<i>Sd</i>
In-home activities	97.98	0	7	1.68	0.91	20	1680	640	320	20	1680	481	353
Working	33.16	0	2	0.38	0.58	10	975	473	144	10	975	436	162
Services	3.87	0	2	0.05	0.24	5	360	98	104	5	360	88	97
Out-of-home eating	8.42	0	3	0.09	0.33	15	300	79	67	15	300	74	67
Grocery shopping	8.42	0	2	0.09	0.29	5	115	30	26	5	115	29	26
Non-daily shopping	6.57	0	1	0.07	0.25	5	290	61	65	5	290	61	65
Education	4.21	0	2	0.05	0.26	50	490	318	147	50	490	266	133
Social activity	12.63	0	2	0.13	0.36	35	770	202	162	32	770	198	164
Leisure	8.59	0	3	0.10	0.33	10	655	191	137	10	655	178	129
Bring/get	11.28	0	5	0.18	0.63	5	205	31	41	5	135	21	30
Touring	0.00	0	0	0.00	0.00	-	-	-	-	-	-	-	-
Other	0.00	0	0	0.00	0.00	-	-	-	-	-	-	-	-
Sleeping	100.00	2	3	2.01	0.09	435	1482	906	119	218	705	451	57

Table 3.1: Descriptive statistics concerning the activity-travel sequences

Figures 3.1 to 3.4 display the histograms with respect to the socio-demographical data underlying the activity-travel sequences. Because these figures are self-explanatory, only the most salient characteristics of these variables are emphasized. The number of working hours is estimated based on the work schedule and is not deduced from the diary data. This variable is only calculated for the individuals who work either part time or full time. This number reaches its highest level around 40 hours, which corresponds to a full time work schedule which is widely applied in Belgium. The histogram displaying the counts of the weekdays shows that the data are more or less evenly distributed along the days of the week, with the largest share of the sequences recorded on Sundays and the smallest on Wednesdays.



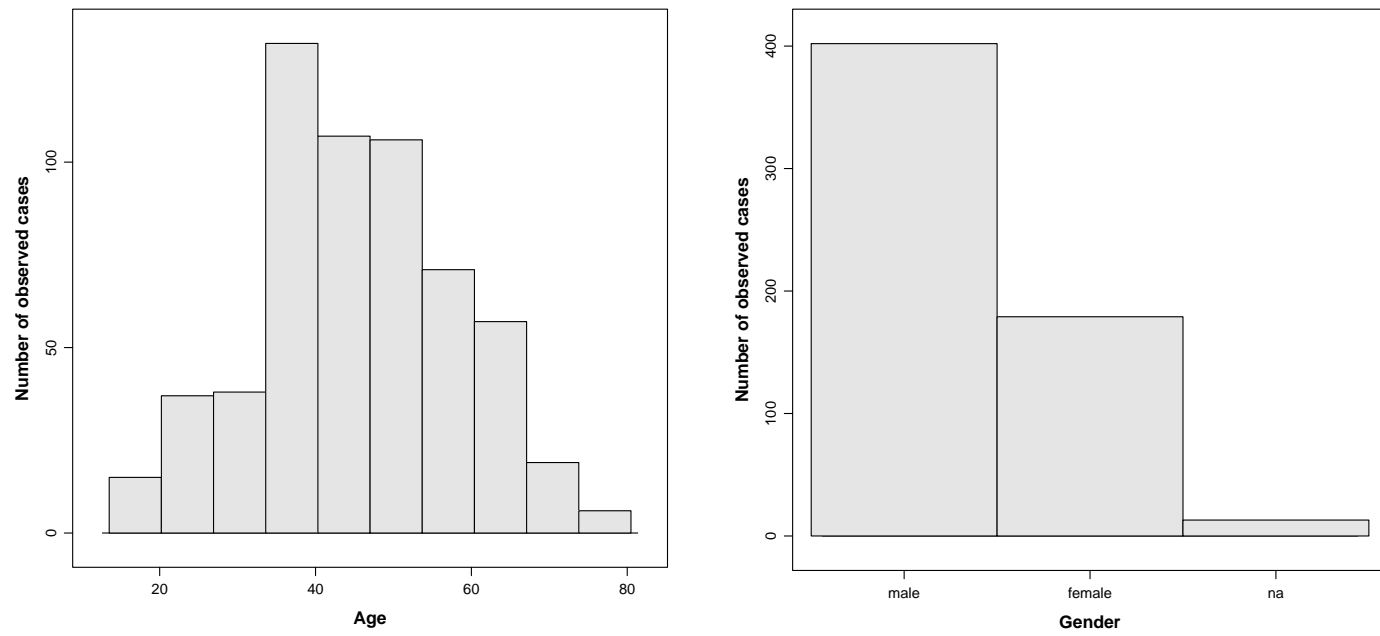


Figure 3.1: Histograms concerning the socio-demographical data: age and gender

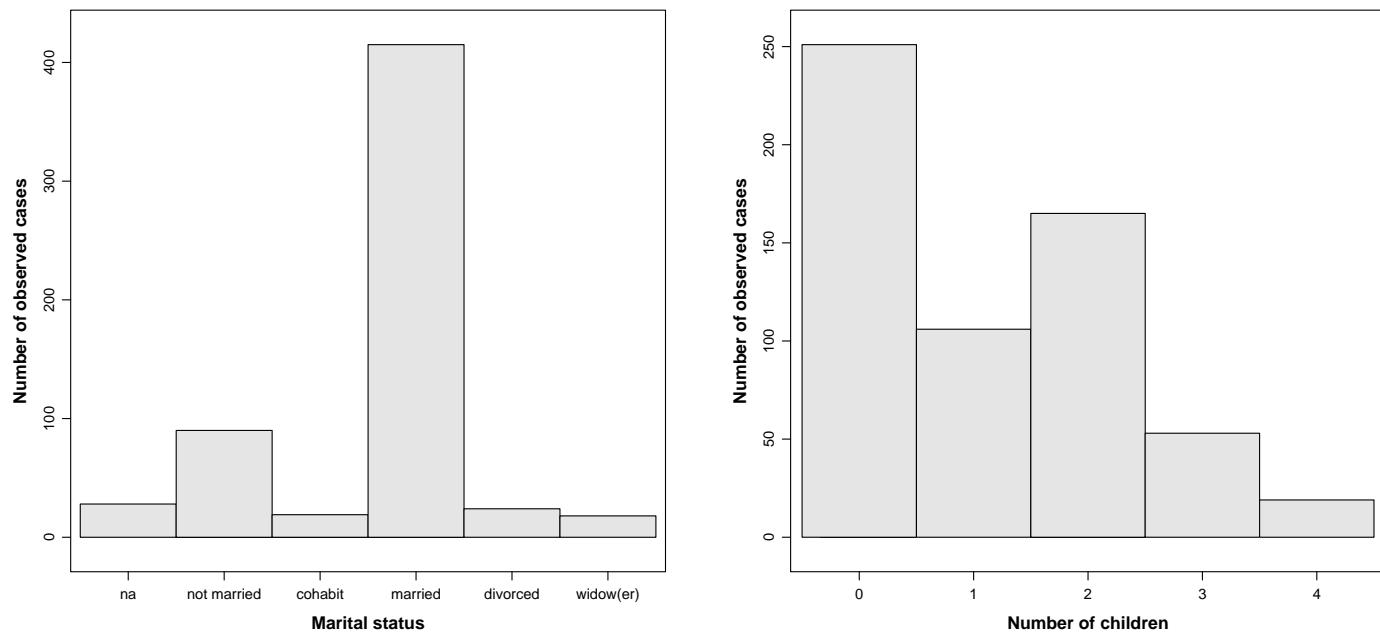


Figure 3.2: Histograms concerning the socio-demographical data: marital status and number of children

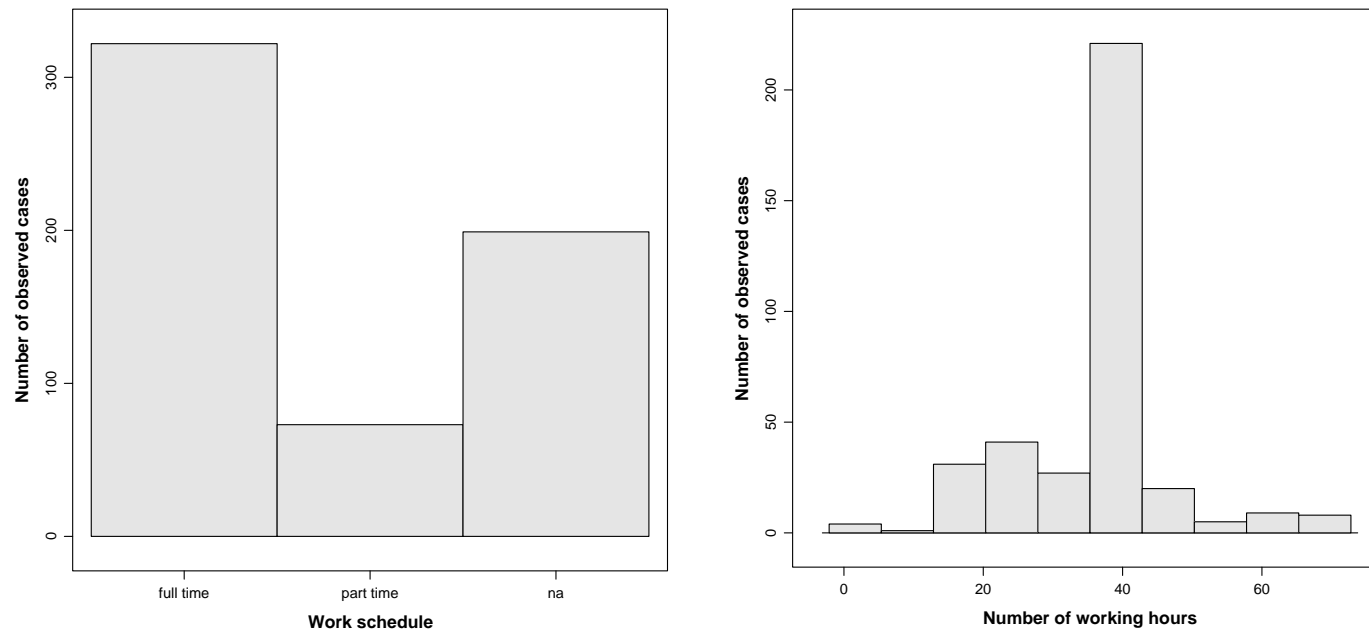


Figure 3.3: Histograms concerning the socio-demographical data: work schedule and number of working hours

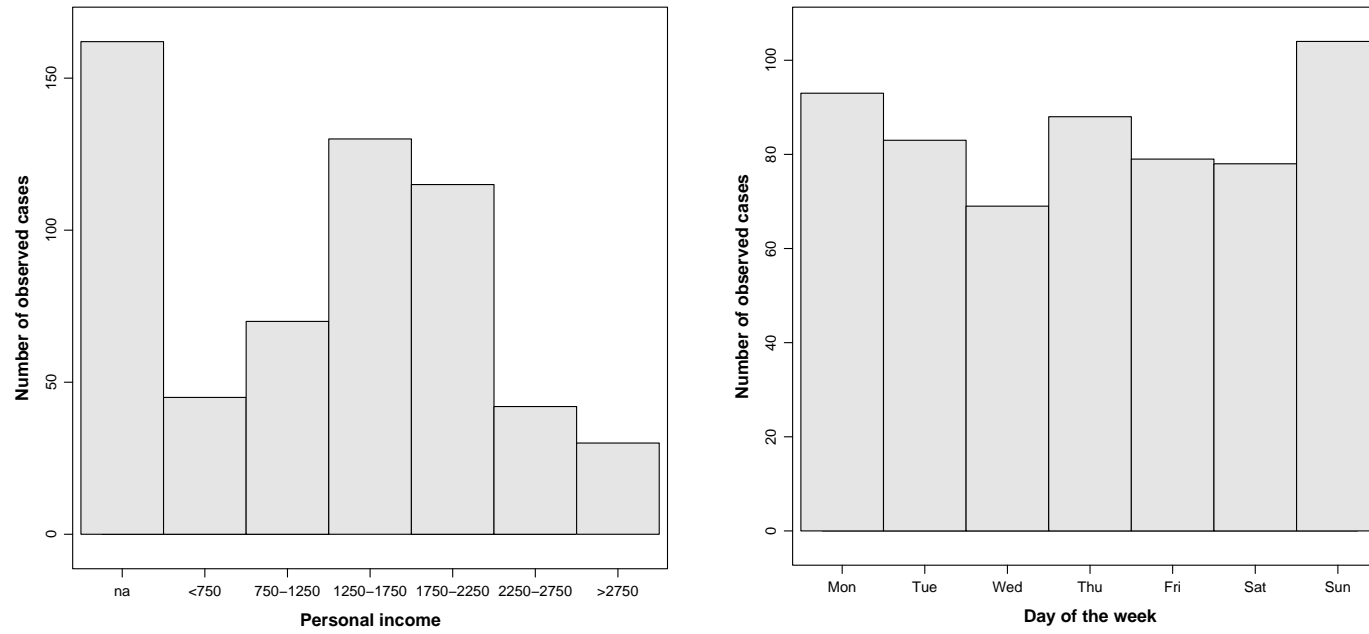


Figure 3.4: Histograms concerning the socio-demographical data: income category and day of the week

### 3.3 Measures of (Dis)similarity

#### 3.3.1 Sequence Alignment Method

As already pointed out in the introduction to this chapter, activity-travel sequences of individuals differ in time and space. After all, in their goal to satisfy their diverse needs and wants, people engage in a number of activities (Chapin, 1974). Yet a number of constraints - such as capability constraints, coupling constraints and authority constraints, also discussed in section 1.2 - restrict people's motivations and opportunities (Hägerstrand, 1970). Consequently, Hägerstrand (1970) argues that time and space are interrelated and have to be accounted for simultaneously. From this point of view, he recognizes the existence of so-called space-time prisms which define both the geographical and temporal boundaries of an individual's activity pattern by accounting for the available time budget, the transport system and perceived choice options, and in which travel enables trading time for distance (Hägerstrand, 1970; Jones, 1979). Bearing this in mind, the observed activity-travel sequences can be converted to Hägerstrand trajectories, which show for each time step - plotted on the  $Z$ -axis - the geographical position of an individual - determined by the  $(x, y)$ -coordinates of the location and plotted in the  $(X, Y)$ -plane. Figure 3.5 illustrates the revealed Hägerstrand trajectories of a random subset of the data described in section 3.2, which are projected on a map of the study area (Flanders, Belgium). The time step in this figure is equal to one minute.

From this perspective, this section intends to reveal space-time dependencies and to analyse differences in the individual space-time paths. To this end, a technique is introduced to compare individuals' activity-travel patterns in space and time. On the one hand, this method enables identifying groups of individuals revealing similar activity-travel sequences and linking socio-demographic profiles to these groups (Wilson, 1998b). On the other hand, this technique can be applied to quantify the fit of predicted activity-travel patterns to observed ones (Joh, 2004).

Several attempts are formulated to grasp differences or similarities between activity-travel sequences. Previous analyses often focus on clustering similar daily activity-travel patterns based on differences in the time and space allocation among subgroups (Kulkarni & McNally,

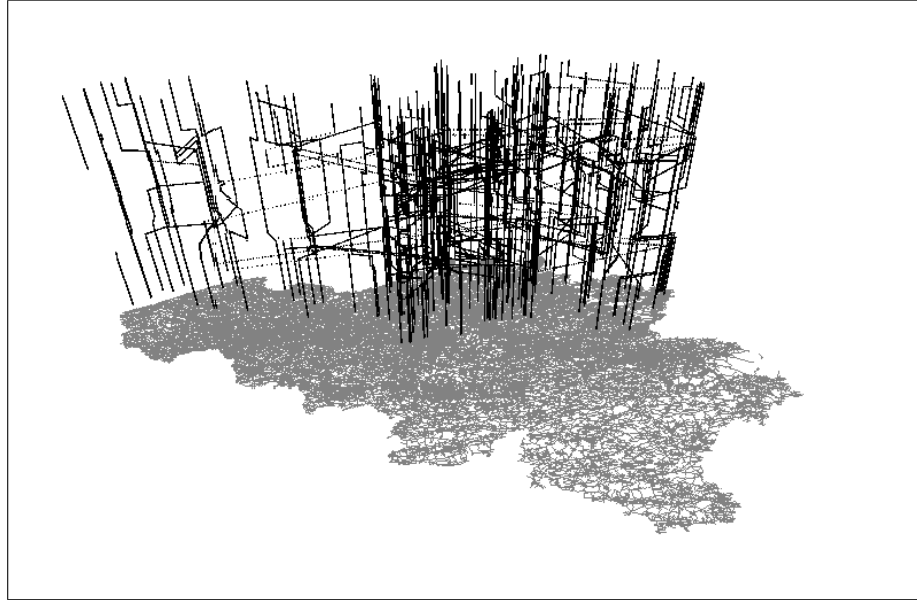


Figure 3.5: Hågerstrand trajectories of a subset of the data projected on a map of the study area

2001; Pas, 1983; Schlich, 2001), while disregarding the relationship between the elements of the sequence (Wilson, 1998a). However recently, efforts studying the sequences as a whole, gain importance (Joh *et al.*, 2001; 2002; 2007; Wilson, 1998a;b; 2001; 2008). These researches are based on a technique originating from the disciplines of computer science and molecular biology and which is called the sequence alignment or string matching method. This method is assumed to define a concept of difference and similarity, which captures the complexity of activity-travel patterns. The goal of the procedure is to equalize (align) two sequences by means of a minimal number of changes in the sequences as explained in the remainder of this paragraph (Wilson, 1998a;b). The outcome of this method is a measure of dissimilarity or distance between two sequences.

For the purpose of explaining the theoretic concept, assume two unidimensional sequences, a source  $S$  and target sequence  $T$ , which have to be aligned. The sequences of activities displayed in figure 3.6 are used to illustrate the functioning of this method. Both sequences consist of a number of episodes or elements  $s_i$  ( $i \in [1, N]$ ) and  $t_j$  ( $j \in [1, M]$ ) where  $N$  and  $M$  are the length of the sequences  $S$  and  $T$  respectively.

$$\begin{aligned}
 S_1 &= \boxed{S \mid H \mid W \mid H \mid S} & N &= 5 \\
 S_2 &= \boxed{S \mid H \mid W \mid G \mid H \mid S} & M &= 6 \\
 T &= \boxed{S \mid H \mid W \mid L \mid H \mid S} & M &= 6
 \end{aligned}$$

Figure 3.6: Examples of unidimensional sequences

Basically, the sequence alignment method aims at transforming the source sequence into the target sequence by means of a number of elementary operations. The elementary operations include either indel or substitution. Indel refers to either deleting an element  $s_i$  from sequence  $S$ , - which corresponds to inserting this element  $s_i$  into the sequence  $T$  - or vice versa. In the example where sequence  $S_1$  is transformed into sequence  $T$ , an activity episode characterized by activity type “L” has to be inserted at position 4 in sequence  $S_1$ . Or the other way round, to transform sequence  $T$  into sequence  $S_1$ , the activity episode denoted by “L” ought to be deleted from  $T$ . Substitution comprises a pair of deletion and insertion: first an element  $s_i$  is deleted from sequence  $S$ , which is followed by inserting the element  $t_j$  into the same sequence  $S$ . In the example, when aligning sequences  $S_2$  and  $T$ , the activity denoted by “G” should be deleted from sequence  $S_2$ , and an activity characterized by activity type “L” should be inserted at this position in the sequence  $S_2$ ; the activity “G” of sequence  $S_2$  is thus substituted by an activity “L” to match sequence  $T$ .

Each of these elementary operations is assigned a penalty  $c$ ; the relative size of which depends on the purpose of the research. The total penalty accumulated to equalize the two sequences reflects the dissimilarity of the two sequences (Wilson, 1998a;b). In case multiple alignments are feasible, the alignment inferring the lowest total penalty is selected.

The actual alignment algorithm is designed as follows. First, an empty  $N \times M$  matrix is created. In this matrix, row  $i$  corresponds to the element  $s_i$  of sequence  $S$ , whereas column  $j$  matches the element  $t_j$  of sequence  $T$ , as is visualized in figure 3.7.

	S	H	W	L	H	S
S						
H						
W						
H						
S						

Figure 3.7: Empty  $N \times M$  matrix for sequence alignment

The matrix is filled out from left to right and from top to bottom, starting at the top left corner ( $i = 1$  and  $j = 1$ ) by means of a recursive matching procedure: the score  $D(i, j)$  for an optimal alignment in cell  $(i, j)$  depends on the cost  $d(s_i, t_j)$  of the match or mismatch between the elements  $s_i$  and  $t_j$  and the optima  $D(i - 1, j)$ ,  $D(i, j - 1)$  and  $D(i - 1, j - 1)$  (if these exist), and can be calculated as follows (Wilson, 2008):

$$\begin{aligned}
 D(i, j) = \min[ & D(i - 1, j - 1) + d(s_i, t_j), \\
 & D(i - 1, j) + c, \\
 & D(i, j - 1) + c].
 \end{aligned}
 \tag{3.1}$$

In the current research area, the cost  $d(s_i, t_j)$  of the (mis)match between  $s_i$  and  $t_j$  is generally set to 0 if  $s_i$  equals  $t_j$  and 1 otherwise; and the penalty  $c$  is equal to 1 (Wilson, 1998a). Figures 3.8 and 3.9 illustrate how the algorithm calculates the sequence alignment method for the example sequences  $S_1$  and  $T$  of figure 3.6, in the course of the alignment process and at the end of this process respectively. The final cost of the optimal alignment is equal to the value  $D(N, M)$ , which indicates the minimal number of elementary operations required to align the sequences.

	S	H	W	L	H	S
S	0	1	2	3	4	4
H	1	0	1	2	2	3
W	1	1	0			
H						
S						

Figure 3.8:  $N \times M$  matrix for sequence alignment in the course of the alignment process

	S	H	W	L	H	S
S	0	1	2	3	4	4
H	1	0	1	2	2	3
W	1	1	0	1	2	2
H	1	1	1	1	1	2
S	0	1	2	2	2	1

Figure 3.9:  $N \times M$  matrix for sequence alignment at the end of the alignment process



### 3.3.2 Multidimensional Dissimilarity Measure

This method provides a valuable measure for determining the dissimilarity between two sequences of activity episodes because it analyses the whole sequence at once. However, in some cases it is not sufficient for a number of reasons. First, the traditional sequence alignment technique only deals with unidimensional patterns, although in the current research area, activity-travel sequences are composed of a number of dimensions (e.g. the activity and the location) which are interrelated (Gärling *et al.*, 1997; Joh *et al.*, 2002). To this end, Joh *et al.* (2002) propose a multidimensional sequence alignment method to meet this limitation. Furthermore, the conventional sequence alignment method is designed in particular to compare sequences containing nominal attributes, such as activity types, but are less suited to analyse sequences which consist of scale values, such as location characteristics (e.g.  $(x, y)$ -coordinates). Figure 3.10 illustrates such multidimensional sequence. The first dimension reflects the activity type, whereas the second dimension records the  $(x, y)$ -coordinates of the corresponding activity locations.

$S$	=	<table style="border-collapse: collapse; text-align: center;"> <tr><td>S</td><td>H</td><td>W</td><td>H</td><td>S</td></tr> <tr><td>(30,40)</td><td>(30,40)</td><td>(60,50)</td><td>(30,40)</td><td>(30,40)</td></tr> </table>	S	H	W	H	S	(30,40)	(30,40)	(60,50)	(30,40)	(30,40)	$N = 5$
S	H	W	H	S									
(30,40)	(30,40)	(60,50)	(30,40)	(30,40)									
$T_1$	=	<table style="border-collapse: collapse; text-align: center;"> <tr><td>S</td><td>H</td><td>W</td><td>H</td><td>S</td></tr> <tr><td>(180,60)</td><td>(180,60)</td><td>(150,50)</td><td>(180,60)</td><td>(180,60)</td></tr> </table>	S	H	W	H	S	(180,60)	(180,60)	(150,50)	(180,60)	(180,60)	$M = 5$
S	H	W	H	S									
(180,60)	(180,60)	(150,50)	(180,60)	(180,60)									
$T_2$	=	<table style="border-collapse: collapse; text-align: center;"> <tr><td>S</td><td>H</td><td>W</td><td>H</td><td>S</td></tr> <tr><td>(30,40)</td><td>(30,40)</td><td>(27.5,35)</td><td>(30,40)</td><td>(30,40)</td></tr> </table>	S	H	W	H	S	(30,40)	(30,40)	(27.5,35)	(30,40)	(30,40)	$M = 5$
S	H	W	H	S									
(30,40)	(30,40)	(27.5,35)	(30,40)	(30,40)									

Figure 3.10: Examples of multidimensional sequences containing the activity type and  $(x, y)$ -coordinates of the activity location

In order to compare sequences based on both the activity type and location, Wilson (2008) extends the basic algorithm discussed in the previous paragraph. To this end, he redefines the unidimensional distance penalty  $d(s_i, t_j)$  to a multidimensional distance penalty, which is equal to the weighted sum of (1) the penalty  $d_a(s_i, t_j)$  of the (mis)match between the activity component of the  $i^{th}$  element of sequence  $S$  and the  $j^{th}$  element of sequence  $T$  on the one hand and (2) the cost  $d_l(s_i, t_j)$  of the (mis)match between the corresponding location components on the other hand.  $d(s_i, t_j)$  is now calculated as follows:

$$d(s_i, t_j) = w_a * d_a(s_i, t_j) + w_l * d_l(s_i, t_j). \quad (3.2)$$

The penalty  $d_a(s_i, t_j)$  of the mismatch of the activity components still equals zero when the activities are identical and one otherwise. The penalty  $d_l(s_i, t_j)$  of the mismatch of the location components are calculated by means of the Euclidean distance between the location of element  $s_i$  and the location of element  $t_j$  based on their  $(x, y)$ -coordinates (Wilson, 2008):

$$d_l(s_i, t_j) = \sqrt{(x_{s_i} - x_{t_j})^2 + (y_{s_i} - y_{t_j})^2}. \quad (3.3)$$

However, the range of the penalty costs calculated based on the activity type and the location, differ arbitrarily, depending on the data. Therefore, the scales of these costs have to be equalized (Wilson, 2008). For this purpose, Wilson (2008) suggests to use the ratio of these ranges to attune the costs of these components.

### 3.3.3 Spatio-Temporal Dissimilarity Measure

Even though this technique is particularly suited to compute similarities of activity-travel patterns in a small area, such as a city or a region, it is less applicable when comparing sequences in a large area, such as a country, because the method disregards similarities within individual activity-travel patterns, which cover different locations and/or are oriented in different directions (Kwan, 2000). Consider for instance, the three sequences recorded in figure 3.10 and visualized in figure 3.11. These sequences all contain five activity episodes, in particular sleep-home-work-home-sleep (S-H-W-H-S). The first sequence is registered by an individual living in the east, and travelling a distance of approximately 30 kilometres to work in the west. The second sequence belongs to an individual whose home is located in the west and whose work location is situated at a distance of more or less 30 kilometres to the east. Finally, the third sequence is recorded by the first individual's neighbour, who also lives in east, but who only travels roughly 5 kilometres to work in the south.

Figure 3.11 now indicates that traditional similarity measures would fail in comparing these geographically dispersed sequences. After all, based on the dissimilarity measure described in the previous paragraph, the first and second sequence proves to be less similar

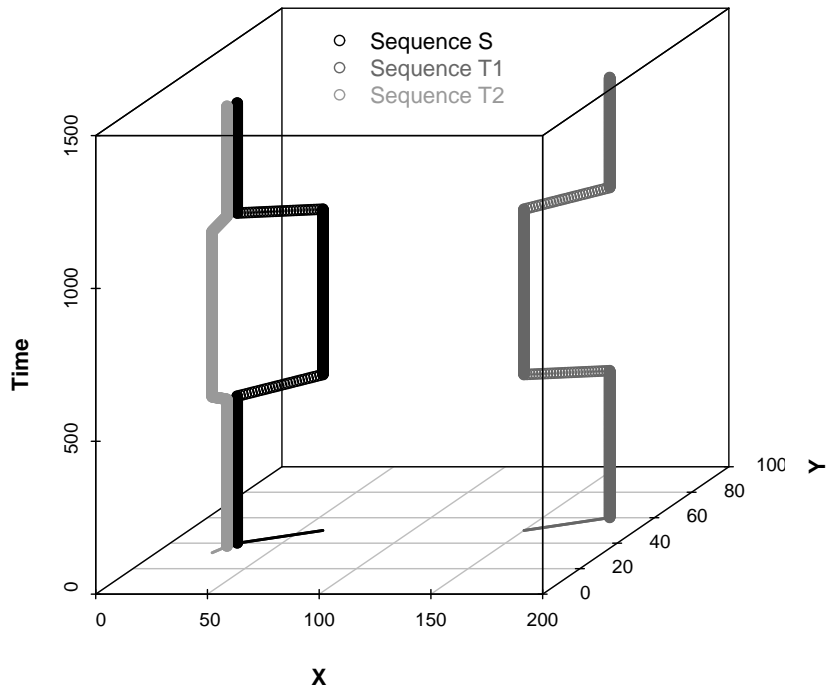


Figure 3.11: Hägerstrand trajectories of example sequences of figure 3.10

than the first and third sequence. However, the only fundamental difference between the two former sequences is the absolute geographical location of the activity locations, while the first individual and the latter one should differ more as their activity sequences do not cover the same area.

These shortcomings can be attributed to two aspects of the method elaborated in the previous paragraph. First, this method calculates the distances between the locations of two sequences rather than accounting for the difference in the distances travelled between the locations *within* each sequence and for the relative position of these locations *within* this sequence. Furthermore, the proposed similarity measure does not capture the fact that one sequence can be geographically rotated and/or translated with respect to the other one. Some simplified cases of these geographical transformations, projected on the  $(X, Y)$ -plane, are illustrated in figure 3.12.

Assuming that the sequences depicted in this figure contain the same sequence of activities (e.g. home-work-shop-leisure) in the same time span, the distance measures between all sequences should equal zero because the relative positions of the locations within these patterns and the distances covered in the course of executing this sequence are identical as well. In this respect, Kwan (2000) proposes to transform or standardize the coordinates of the locations based on the coordinates of the home and work location. In her analysis the home location is situated at the origin  $(0, 0)$  and the home-work axis is rotated so as to become the positive  $X$ -axis and the corresponding plane to become the home-work plane.

This transformation enables revealing valuable information contained in activity-travel sequences, as it eliminates the influence of the absolute geographical position of the locations within a sequence, as well as the effect of the orientation of the sequence on the analysis of dissimilarities between patterns. Yet, one major drawback of this solution is that a number of these patterns do not include a working activity, and can thus not be transformed according to the approach introduced by Kwan (2000).

Given the inadequacy of the existing techniques in comparing geographically dispersed activity-travel patterns, an improvement of the multidimensional sequence alignment method introduced above is proposed here. The purpose of this novel technique entails meeting the limitations of the existing techniques, while being able to account for the multiple dimensions

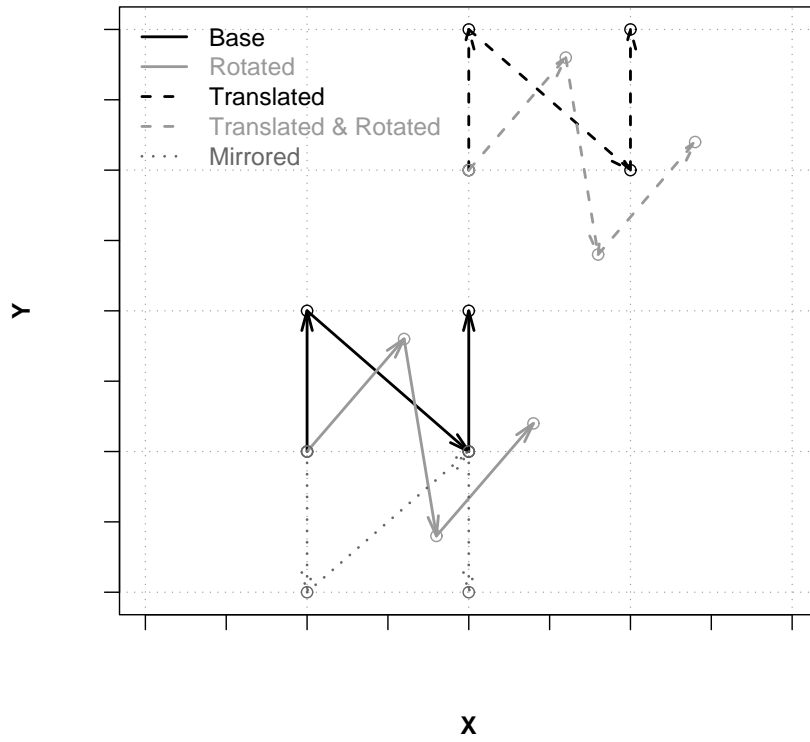


Figure 3.12: Sequences: rotated, translated and mirrored

included in activity-travel sequences. Summarized, the technique first transforms and standardizes the geographical coordinates to reflect the relative geographical movements within the sequences. Subsequently, the multidimensional sequence alignment method introduced by Wilson (2008) is utilized to estimate the dissimilarity between the sequences, considering both the relative geographical movements within the sequences and the remaining dimensions which can be categorical, for example the activity type.

The transformation of the geographical coordinates to attributes describing the relative movements is based on the method described in Vlachos *et al.* (2004). The authors consider a sequence  $S$  to be a trajectory of  $(N - 1)$  movement vectors  $V_{s_i}$ , each determined by the spatial coordinates  $(x_{s_{i-1}}, y_{s_{i-1}})$  and  $(x_{s_i}, y_{s_i})$ , as shown in Figure 3.13.

$S$	=	(0, 0)	(0, 1)	(1, 0)	(1, 1)
$V$	=	[[0, 0)(0, 1)]   [(0, 1)(1, 0)]   [(1, 0)(1, 1)]			
$\hat{V}$	=	$-\frac{\pi}{2}$	$\frac{\pi}{4}$	$-\frac{\pi}{2}$	
$\hat{V}_{Normalized}$	=	$-\frac{\pi}{4}$	$\frac{\pi}{2}$	$-\frac{\pi}{4}$	
$\ \hat{V}\ $	=	1	1.4142	1	

Figure 3.13: Calculating the trajectory of AAL-pairs for the base sequence visualised in figure 3.12

For such vectors  $V_{s_i}$  the rotation angle  $\hat{V}_{s_i}$  with respect to a reference movement vector  $V_{ref}$ , which is set to the positive  $X$ -axis, is calculated. This angle is a number in the range between  $-\pi$  and  $\pi$ .

$$\hat{V}_{s_i} = \text{acos}\left(\frac{x_{s_i} - x_{s_{i-1}}}{\sqrt{(x_{s_i} - x_{s_{i-1}})^2 + (y_{s_i} - y_{s_{i-1}})^2}}\right). \quad (3.4)$$

Next to this rotation angle, the Euclidean length of the arc  $\|\hat{V}_{s_i}\|$  is computed as well and equals:

$$\|\hat{V}_{s_i}\| = \sqrt{(x_{s_i} - x_{s_{i-1}})^2 + (y_{s_i} - y_{s_{i-1}})^2}. \quad (3.5)$$

As a result the spatial coordinates of a sequence are transformed into a trajectory of

Angle/Arc Length pairs (AAL). In order to become a rotation invariant transformation, Vlachos *et al.* (2004) propose to normalize the angles values. To this purpose, the average angle value is subtracted from all values within the trajectory of angles. To ensure that the normalized value falls within the range of  $[-\pi, \pi]$ , the resulting angle values are wrapped within this range by subtracting or adding  $2\pi$  if necessary. Figure 3.13 illustrates the calculation of the AAL-pairs for the base sequences visualized in figure 3.12, while figure 3.14 shows the transformation of the patterns displayed in figure 3.12 to their corresponding angles and subsequently normalized angles. For episode  $s_i$ , the latter figure shows the value  $\hat{V}$  of the angle determined by  $S_{i-1}$  and  $S_i$ . The corresponding arc lengths are not displayed as these are equal for all of the sequences due to the equal distances between the activity locations.

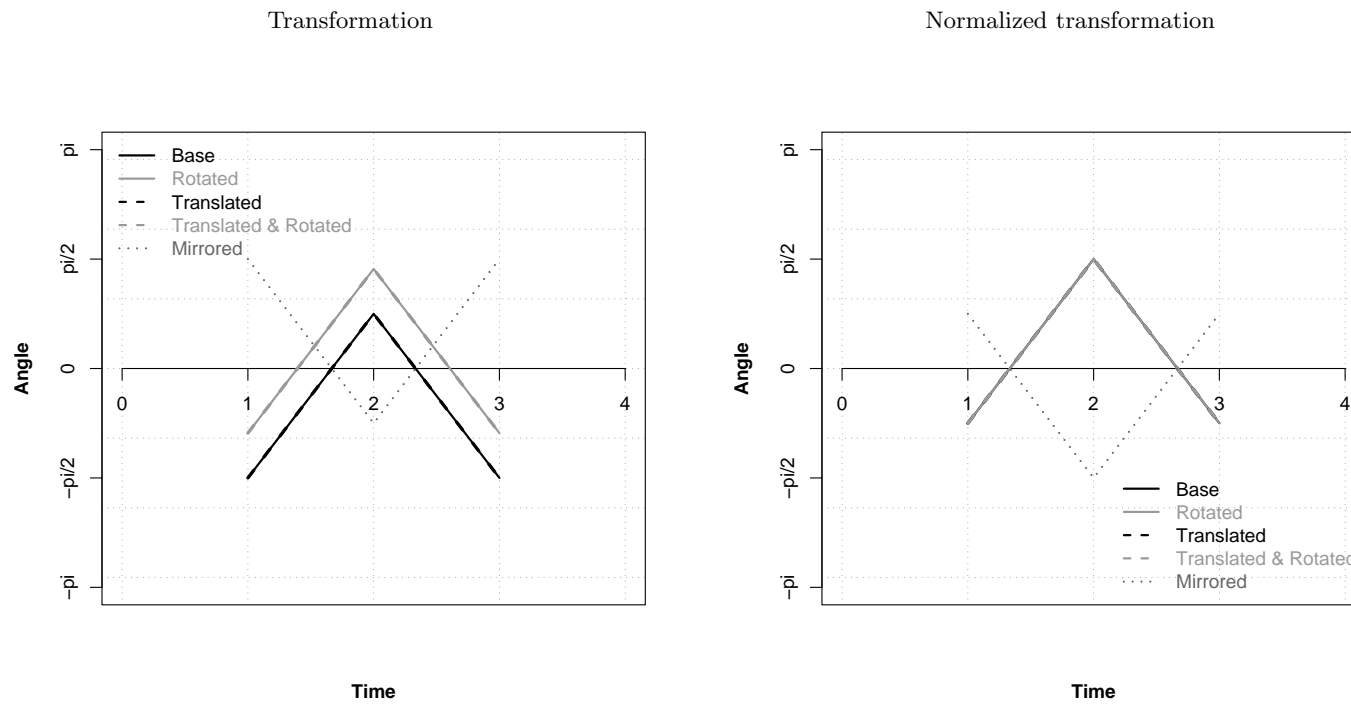


Figure 3.14: (Normalized) trajectories of angles of the sequences of figure 3.12 compared to the (normalized) trajectory of angles of the base sequence



Figure 3.14 reveals that the trajectories of the normalized angles of the rotated and/or translated sequences are the same as the trajectory of the base sequence. This signifies that a distance measure calculated based on the AAL-pairs are invariant to both translation and rotation of activity-travel sequences. However, in the case of the mirrored sequence, the figure indicates that the AAL-trajectories do not run parallel. To solve this issue, the normalized AAL-values have to be mirrored along the  $X$ -axis by multiplying all angles by  $-1$ , as shown in figure 3.15

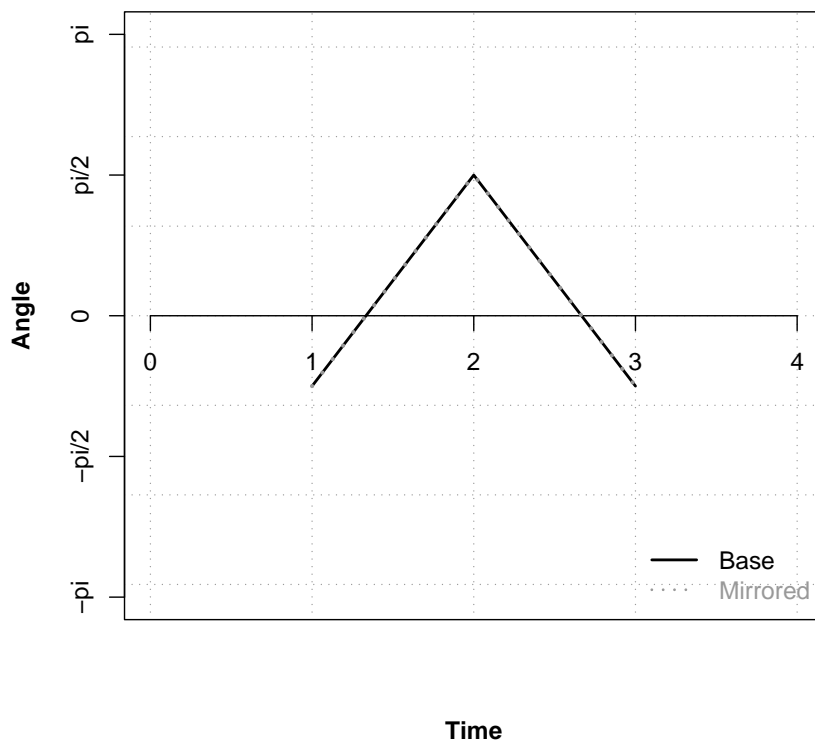


Figure 3.15: Mirrored trajectory of normalized angles of the mirror sequences of figure 3.12 compared to the trajectory of normalized angles of the base sequence

Although this approach seems to be able to meet its objective, in a number of cases the method fails. Some of these “problematic” sequences are illustrated in figure 3.16, next to

their corresponding (normalized) trajectories of angles in figure 3.17. Figure 3.16 also sets these sequences side by side to the sequence matching the normalized trajectory of angles. This figure shows that the overall rotation angles of these problematic sequences do not fall within the range of  $[-\pi/2, \pi/2]$  as is assumed in Vlachos *et al.* (2004).

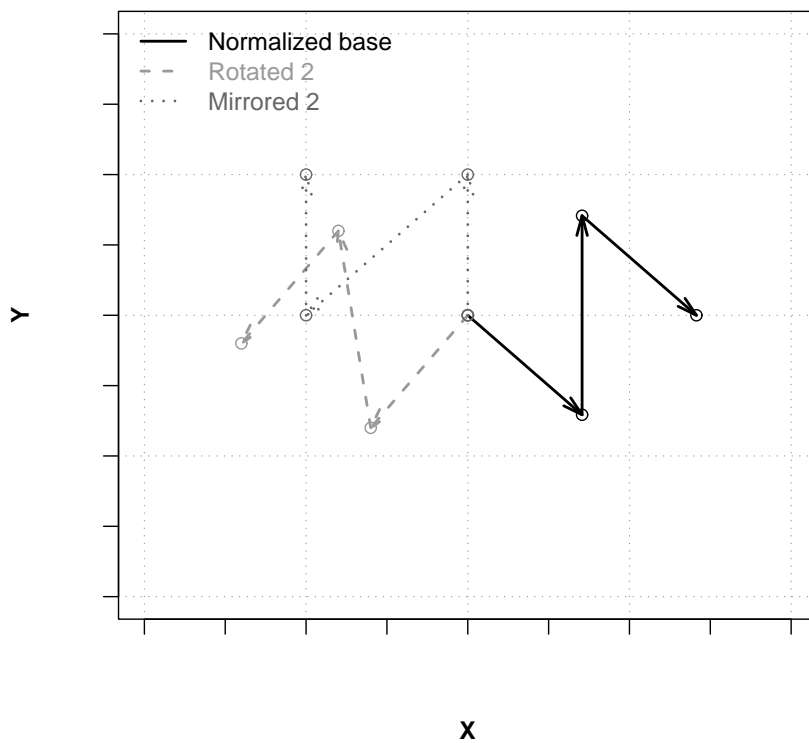


Figure 3.16: Sequences 2: rotated and mirrored

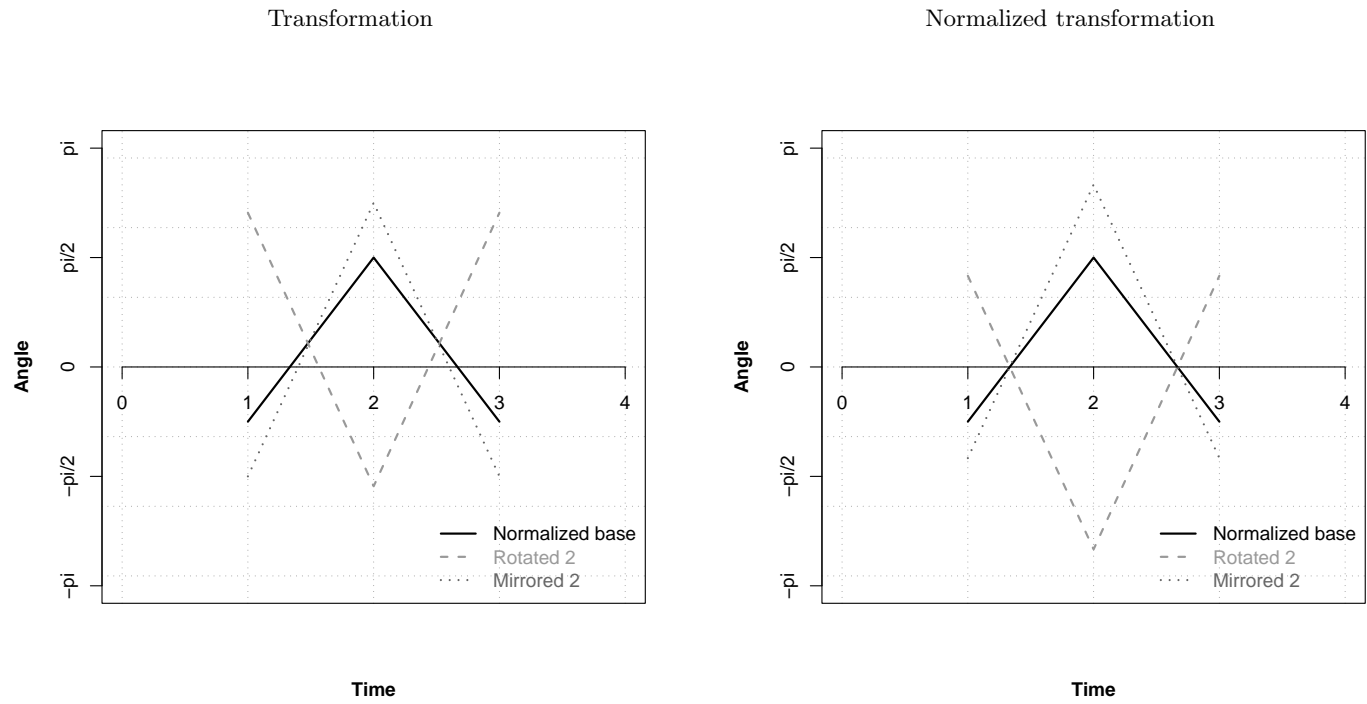


Figure 3.17: (Normalized) trajectories of angles of the sequences of figure 3.16 compared to the (normalized) trajectory of angles of the base sequence

Consequently, the trajectories of angles of these sequences are not comparable to the base sequence. To tackle this problem, the trajectories of angles - either normalized or not - are corrected by subtracting  $\pi$ , wrapping the resulting angles in the range  $[-\pi, \pi]$  and normalizing these angles. The corrected (normalized) trajectories of angles are displayed in figure 3.18. The corrected trajectories reveal the same pattern as the base trajectory, indicating that the underlying sequences are similar.

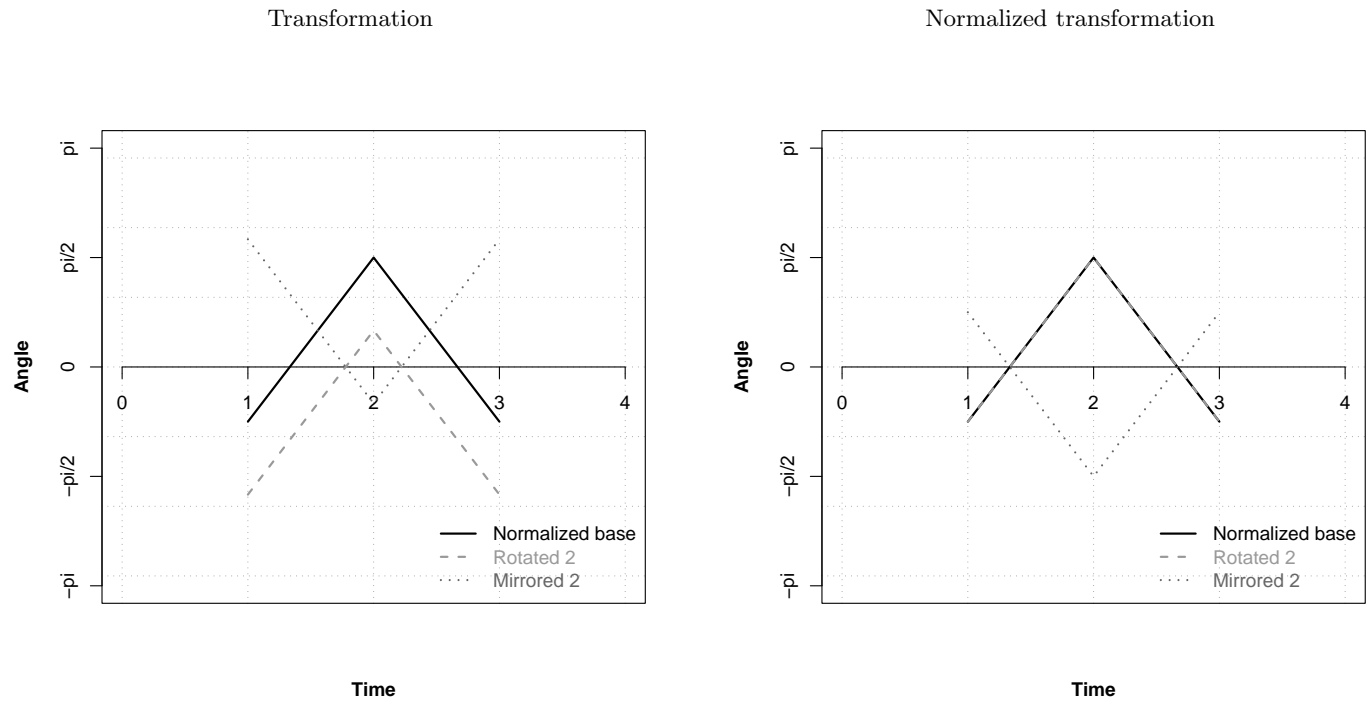


Figure 3.18: Corrected (normalized) trajectories of angles of the sequences of figure 3.16 compared to the (normalized) trajectory of angles of the base sequence

The resulting AAL-patterns thus form the basis of the estimation of the geographical dissimilarity designed here. Therefore, a dissimilarity score needs to be defined: the dissimilarity of two AAL-values  $d_{aal}(aal_1, aal_2)$  equals the weighted sum of the distance between the arc lengths and the distance between the angles. To equalize the ranges of these two distance values, the arc length distance is divided by the maximum arc length observed within the data, and the angle distance is divided by  $2\pi$ .

$$d_{aal}(aal_1, aal_2) = w_{arc} * \frac{d_{arc}(aal_1, aal_2)}{\max(arc\ length)} + w_{angle} * \frac{d_{angle}(aal_1, aal_2)}{2\pi}. \quad (3.6)$$

The distance of two arc lengths is computed by the Euclidean distance, while the distance of two angles  $a_1$  and  $a_2$  is defined as the minimum of  $|a_1 - a_2|$  and  $2\pi - |a_1 - a_2|$ . This penalty cost  $d_{aal}$  replaces the distance measure of the location component in Equation 3.2. The alignment of the sequences is thus based on the AAL's instead of the geographical coordinates. Nevertheless, the actual alignment method remains unaltered. It should be noted that this computational process from defining the AALs to determining the dissimilarity score is entirely automated, and no manual intervention or judgement is required.

Figure 3.19 illustrates one of the sequences shown in figure 3.5 next to its normalized transformation, which is reconstructed based on the corresponding normalized AAL-trajectory displayed in figure 3.20. In the latter figure, the distance from the home location is added for the sake of clarity to show the movements within the pattern, and does not influence the calculation of the AAL-trajectories as such. The figure indicates that in the course of the activity execution, the location of the episodes does not change, causing the angles and arc length to equal zero. Yet, when travelling from one activity location to the next one, the angles and arc lengths differ from zero. In order to compose the normalized transformed sequence of figure 3.19, it is assumed that the first episode is executed at the location with coordinates  $(0, 0)$ .

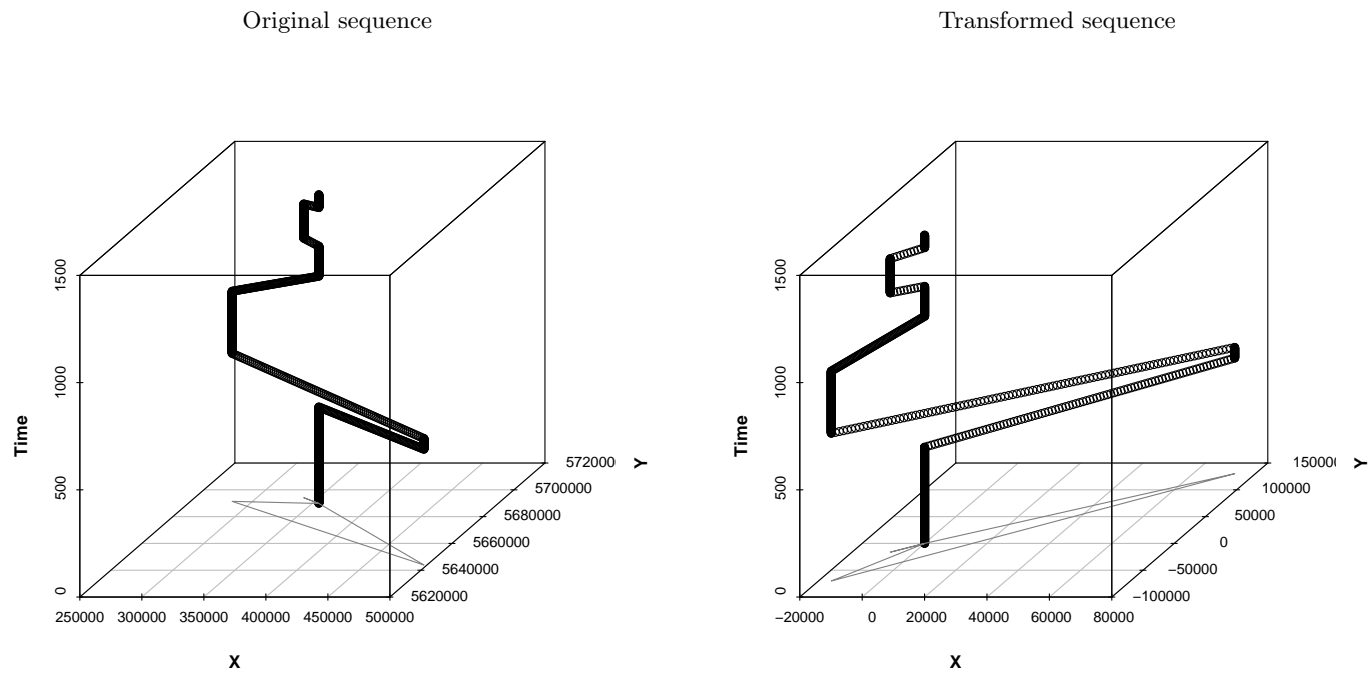


Figure 3.19: Hägerstrand trajectory of one of the sequences

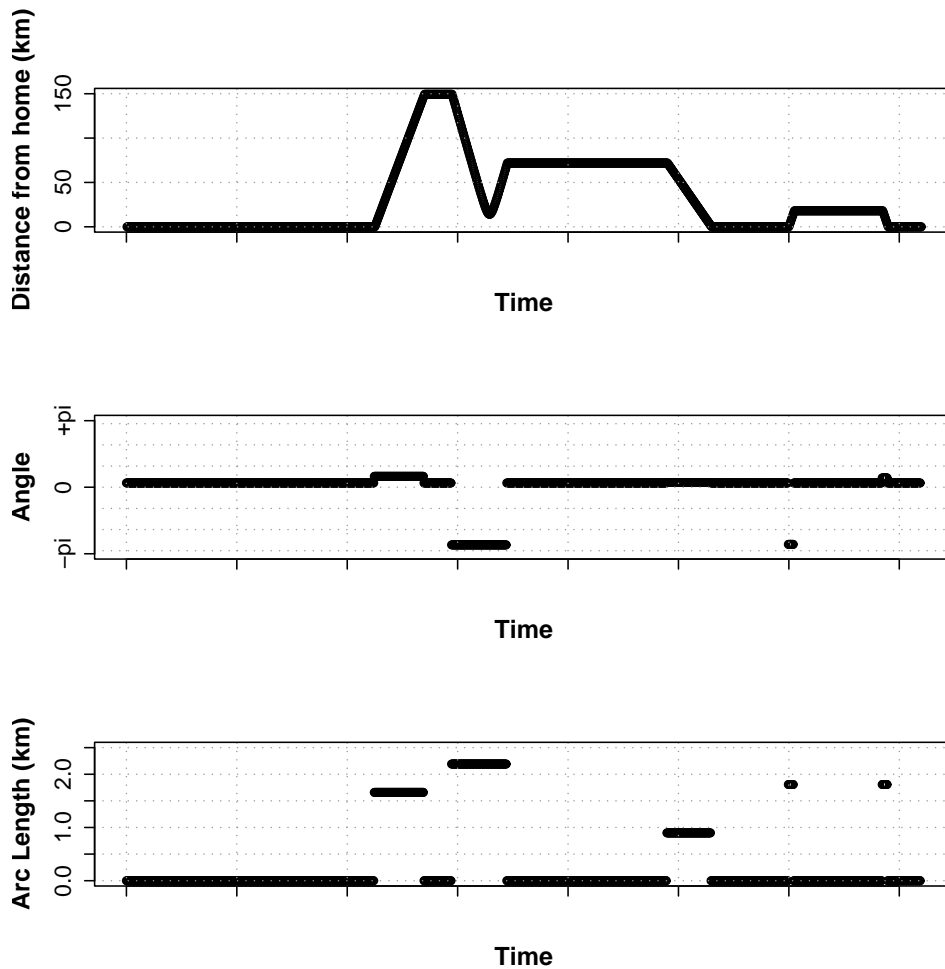


Figure 3.20: Normalized AAL-trajectory of sequence displayed in figure 3.19

The approach presented here has much ground in common with other researches. For instance, Vlachos *et al.* (2004) aspires at capturing similarities of two-dimensional motion patterns (e.g. handwriting data) which can be oriented differently. For this purpose, Vlachos *et al.* (2004) introduced the AAL-transformation discussed above, followed by Dynamic Time Warping to estimate the actual (dis)similarity. Next to this research, McIntosh & Yuan (2005) aims at assessing the similarity of geographic processes and events (e.g. rainfall data), which are described by number of indices - such as the relative movement, the growth, the orientation with respect to the direction of the movement, the size, the granularity of the change -, determining the spatio-temporal characteristics of these sequences and which serve



as basis for the spatio-temporal comparison. (Dis)similarities between these events are also estimated by means of Dynamic Time Warping. Yet, the differences of the current approach with respect to these studies are three-fold. Firstly, the proposed technique does not require calculating geographical indices ahead. The comparison of the activity-travel sequences is based directly on the recorded activity-travel data. Secondly, next to the two-dimensional geographic data of the activity locations, the dissimilarity measure needs to incorporate the remaining dimensions of activity-travel data as well. In particular, the activity type is in any case included in the comparison of activity-travel patterns. The activity type contains a categorical variable, and cannot be processed using Dynamic Time Warping. However, applying an adapted sequence alignment method to calculate the dissimilarity of activity-travel sequences meets this requirement. Thirdly, the rotation invariant distance measure forwarded in Vlachos *et al.* (2004) disregards similarities between sequences with a mutual rotation outside the interval  $[-\pi/2, \pi/2]$ . Though efficient when comparing handwriting data (e.g. the number 6 should not be similar to the number 9), this assumption restricts the usefulness of the approach in the current research area. Therefore, the proposed approach elaborates on the suggested transformation of the geographical coordinates to enable comparing sequences which are rotated up to  $2\pi$  with respect to one another.

### 3.4 Identification of Groups of Similar Behaviour

#### 3.4.1 Clustering

As already pointed out in the introduction to the previous section, the purpose of the technique described above includes identifying groups of individuals who show strong resemblance in their Hagerstrand trajectories. To reach this aim, first the mutual distances between sequences present in the data are calculated based on the proposed spatio-temporal dissimilarity measure. The sequences can be processed either in the so-called short or in the long form. The short form describes activity-travel sequences by the activity and location attributes of the episodes, while a sequence in the long form consists of a number of time slots, each of which represent  $t$  minutes within a day and which are characterized by an activity type and a location. The long form enables incorporating the activity duration into

the distance measure (Wilson, 1998b). The difference between these formats is illustrated in figure 3.21 for the example sequence  $S$  of figure 3.10, and in figure 3.22 for the sequence recorded in figure 3.19.

$$S_{short} = \boxed{S} \boxed{H} \boxed{W} \boxed{H} \boxed{S} \quad N = 5$$

$$S_{long} = \boxed{S} \boxed{S} \boxed{S} \boxed{S} \boxed{S} \boxed{S} \boxed{S} \boxed{S} \boxed{H} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{W} \boxed{H} \boxed{H} \boxed{H} \boxed{H} \boxed{H} \boxed{S} \quad N = 24$$

Figure 3.21: Sequence represented in the long form (time slots of 1 hour) vs. the short form for the example sequence  $S$  of figure 3.10

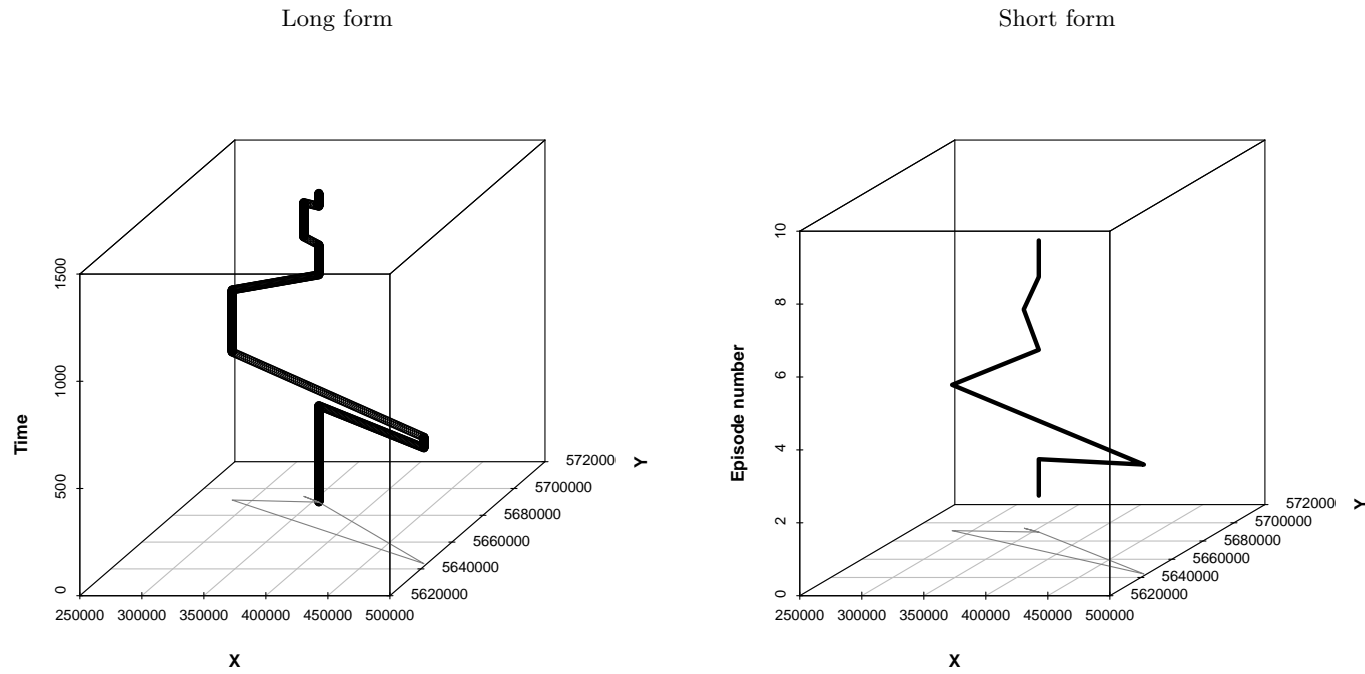


Figure 3.22: Hagerstrand trajectories of sequences represented in the long form (time slots of 1 minute) vs. the short form for the sequence recorded in figure 3.19

In the current research the size  $t$  of the time slots equals one, causing each sequence in the long format to contain 1440 episodes. For the purpose of clarity, the results based on the sequences in the short form are enclosed in the Appendix A.

After having calculated the mutual dissimilarity scores, these values are fed into a clustering algorithm dividing the sequences into homogeneous groups. Because the aim of the current research consists of finding the best classification of the data into a number of groups, a partitioning algorithm is preferred to a hierarchical clustering method (Kaufman & Rousseeuw, 1990). Furthermore, the clustering technique selected here has to be able to accept a predefined dissimilarity table as input, because recalculating dissimilarities in the course of the clustering process - as is the case for the well-known  $k$ -means clustering method - is not feasible. The basic clustering approach *pam* (Partitioning Around Medoids) satisfies these requirements Kaufman & Rousseeuw (1990), and therefore is applied here. Furthermore, the *pam* method is supplied by *cluster* package implemented in *R*, a free software environment for statistical computing and graphics. More details on the selected clustering algorithm can be found in Kaufman & Rousseeuw (1990).

In order to determine the number of clusters  $k$ , the *pam* clustering algorithm is run several times, while varying the number of clusters  $k$ . The number of clusters is set based on the average silhouette width for the entire data set. This measure is provided by the algorithm and reflects the quality of the clusters: the average silhouette width approaches one if the clustering enables revealing a strong structure, while an average silhouette width of  $-1$  signifies that the clustering does not disclose a substantial structure. The average silhouette width should exceed 0.25 at least (and preferably even 0.50) if a structure is revealed within the data (Kaufman & Rousseeuw, 1990). Based on this knowledge and the findings summarized in table 3.2 and figure 3.23, the number of clusters is fixed to 3.

### 3.4.2 Validation of the Spatio-Temporal Dissimilarity Measure

The cluster results enable indicating the validity of the distance measure developed in section 3.3.3. To this end, the existing distance measure described in section 3.3.2 is applied to the data as well, and is used to cluster the sequences into groups displaying similar spatial and temporal behaviour. The number of clusters based on these dissimilarity scores is also set

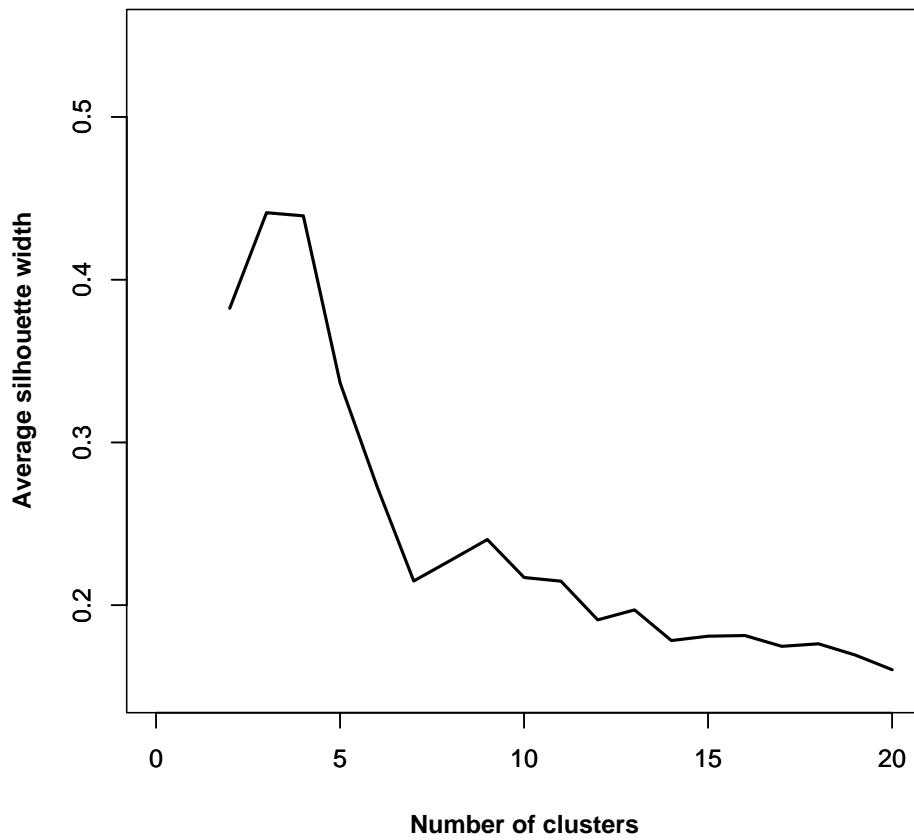


Figure 3.23: Cluster results for varying number of clusters  $k$  based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3

$k$	Avg sil. width	Min. #	Max. #
2	0.3824	192	402
3	0.4411	92	313
4	0.4392	17	299
5	0.3366	17	242
6	0.2732	17	179
7	0.2149	17	176
8	0.2275	16	176
9	0.2404	12	174
10	0.2170	12	174
11	0.2148	11	174
12	0.1910	11	129
13	0.1971	11	128
14	0.1783	11	128
15	0.1810	11	112
16	0.1814	11	111
17	0.1748	11	111
18	0.1763	6	109
19	0.1694	6	109
20	0.1604	6	85

Table 3.2: Average silhouette width for varying number of clusters  $k$  based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3

to 3 as shown in table 3.3 and figure 3.24.

The actual validation of the proposed distance measure occurs based on a number of figures describing some features of the sequences classified in each of the clusters. The first set of statistics to be calculated are introduced by Lefever (1926), adapted by Bachi (1963) and Yuill (1971) and implemented by Buliung & Remmel (2008) in the *aspace* toolkit available within the statistical package *R*. This toolkit aims at visualizing and describing spatial properties of individual activity spaces by establishing an approach to quantify the centrality and dispersion of the activity sequences in space. To reach this goal, the *aspace* toolkit includes two centographic statistical functions, computing either the standard distance deviation (SDD) (Bachi, 1963) or the standard deviation ellipse (SDE) (Yuill, 1971). The SDD-value reflects the standard deviation of the distances between the activity locations and a centre location which can be defined either exogenously by the user or which can be estimated by the algorithm. This SDD value equals the radius of a circle for which the centre is set to a predetermined or estimated centre location and is interpreted as the dispersion of the activity locations with respect to this centre location. For the sequence presented in figure

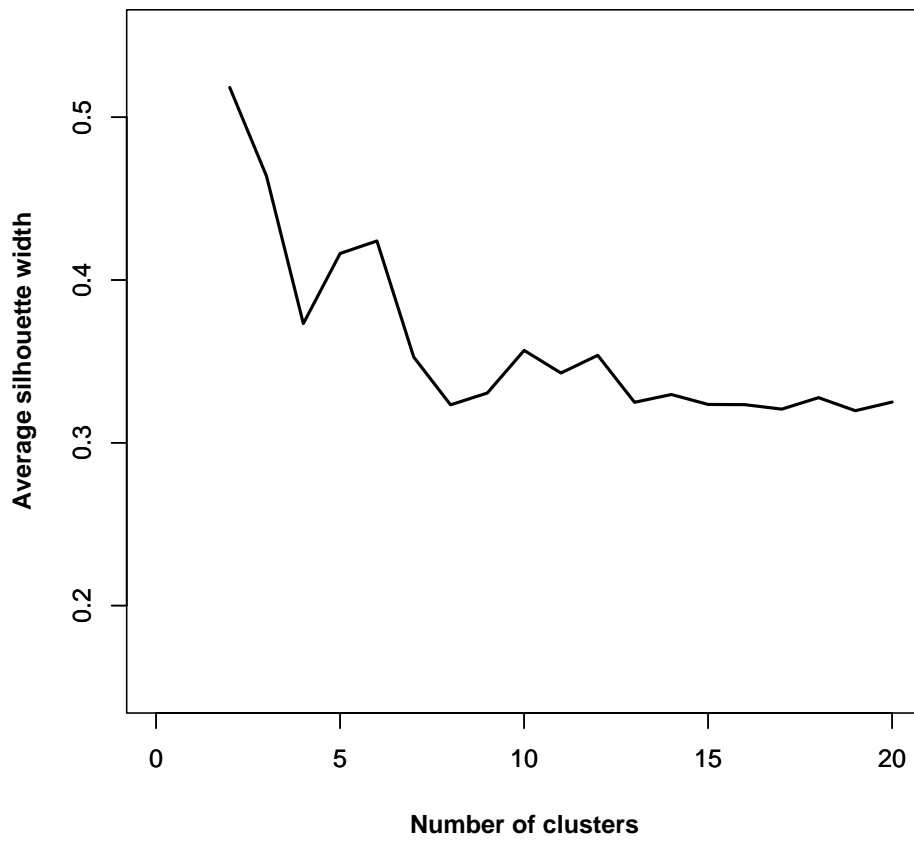


Figure 3.24: Cluster results for varying number of clusters  $k$  based on the existing distance measure described in section 3.3.2

<i>k</i>	Avg sil. width	Min. #	Max. #
2	0.5183	283	311
3	0.4638	131	248
4	0.3732	130	187
5	0.4163	48	146
6	0.4240	38	146
7	0.3527	38	137
8	0.3233	38	122
9	0.3306	32	101
10	0.3568	18	111
11	0.3429	18	111
12	0.3537	18	111
13	0.3249	17	80
14	0.3297	15	70
15	0.3236	13	66
16	0.3234	10	66
17	0.3207	10	68
18	0.3277	10	68
19	0.3197	10	68
20	0.3250	10	68

Table 3.3: Average silhouette width for varying number of clusters  $k$  based on the existing distance measure described in section 3.3.2

3.19, this concept is illustrated in figure 3.25.

However, the presence of spatial outliers within an activity sequence proves to bias the SDD measure. Therefore, the SDE measure is established, which expresses the dispersion and directional bias of spatial sequences without being as distracted by spatial outliers compared to the SDD measure. The SDE method hypothesizes that the spatial dispersion of the activity locations can be characterized as a rotated ellipse with respect to a predetermined or estimated centre location, rather than a circle. This SDE-value for the sequence of figure 3.19 is displayed in figure 3.26.

In the presented research, the SDE measure is calculated for each of the activity-travel sequences, assuming the centre location equals the home location. Bearing in mind the main focus of this study, the eccentricity and the area of the ellipse resulting from the SDE calculations are recorded because these parameters indicate the shape and the size of the ellipse respectively. An eccentricity approaching one, signifies an elongated ellipse, while an eccentricity of zero indicates a circle, in which the minor axis equals the major axis (Buliung & Rimmel, 2008). Next to these spatial descriptive statistics, the main geographical position



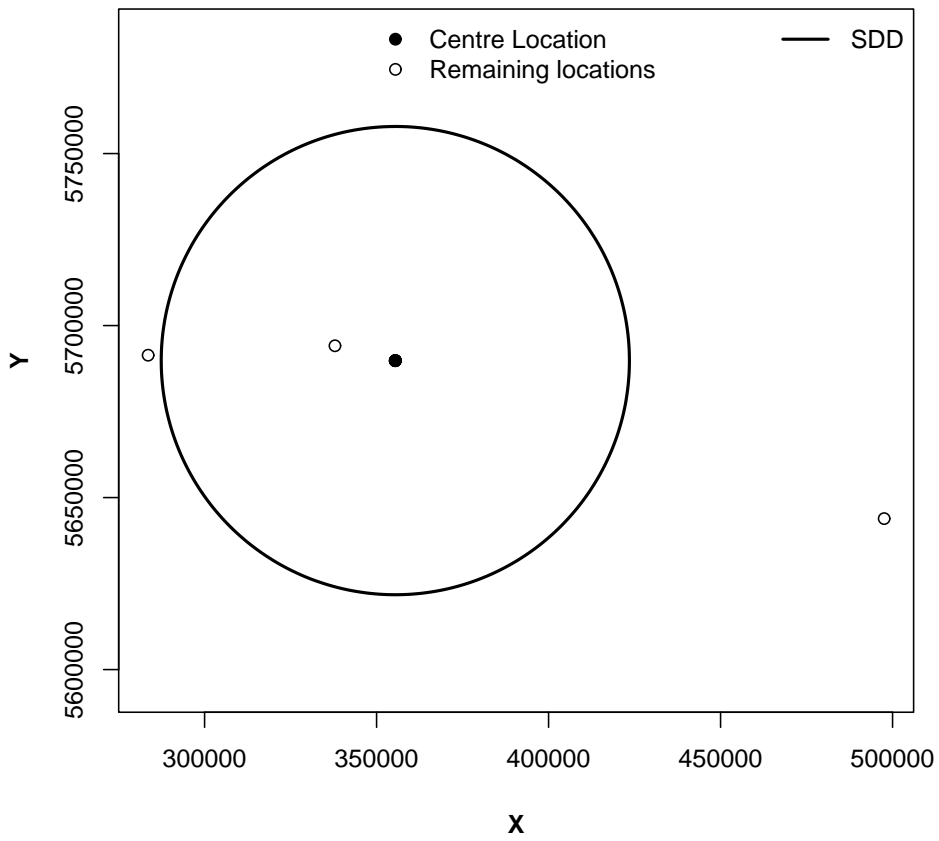


Figure 3.25: Determining the SDD-value for the observed sequence visualised in figure 3.19

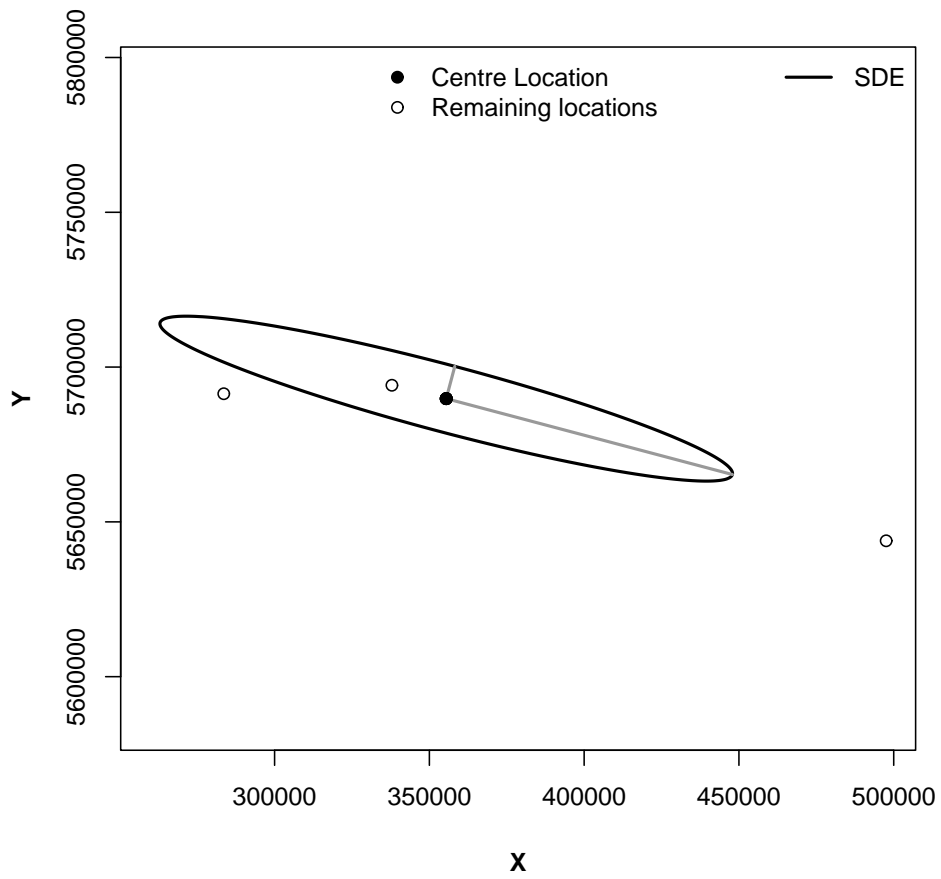


Figure 3.26: Determining the SDE-value for the observed sequence visualised in figure 3.19

---

### 3.4. IDENTIFICATION OF GROUPS OF SIMILAR BEHAVIOUR

---

of the sequence is determined, founded on the average  $(x, y)$ -coordinates weighted according to the duration spent at each location.

	Cluster 1	Cluster 2	Cluster 3	All
# of sequences	189	313	92	<b>594</b>
# of episodes per sequence				
Avg	5.91	4.60	3.30	<b>4.82</b>
Sd	1.39	2.24	0.69	<b>2.02</b>
Eccentricity				
Avg	0.9909	0.4905	0.3260	<b>0.6242</b>
Sd	0.0781	0.4993	0.4713	<b>0.4830</b>
Area of SDE ( $km^2$ )				
Avg	128	11	13	<b>49</b>
Sd	646	76	129	<b>375</b>
Weighted Avg x ( $km$ )				
Avg	375	362	376	<b>368</b>
Sd	62	76	69	<b>71</b>
Weighted Avg y ( $km$ )				
Avg	5,668	5,670	5,671	<b>5,670</b>
Sd	17	17	17	<b>17</b>

Table 3.4: Descriptive statistics of cluster results based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3

Tables 3.4 and 3.5 summarize these statistics based on the clustering performed on both distance measures. The numbers in these tables already lift the veil on the potential of both methods and the differences between these approaches. While the values of the average and standard deviation of the eccentricity and area of SDE show that the groups defined based on the proposed spatio-temporal dissimilarity measure are largely distinct with respect to the criteria of the extent of spatial dispersion, these differences are less explicit in case of the cluster results obtained by the existing distance measure. The other way round, the clusters distinguished based on the proposed spatio-temporal dissimilarity measure display a larger geographical spread as reflected in the weighted average  $(x, y)$ -coordinates, than the clusters formulated based on the existing distance measure. What strikes one most is the relatively low standard deviation of the weighted average  $(x, y)$ -coordinates of the latter cluster results, compared to the former cluster results, underpinning this finding.

However, these conclusions should be supported formally. For that purpose, ANOVA is

	Cluster 1	Cluster 2	Cluster 3	All
# of sequences	131	215	248	<b>594</b>
# of episodes per sequence				
Avg	4.70	4.76	4.93	<b>4.82</b>
St. dev	1.74	2.07	2.12	<b>2.02</b>
Eccentricity				
Avg	0.6060	0.6032	0.6521	<b>0.6242</b>
Sd	0.4870	0.4890	0.4761	<b>0.4830</b>
Area of SDE ( $km^2$ )				
Avg	87	46	31	<b>49</b>
Sd	692	264	156	<b>375</b>
Weighted Avg x ( $km$ )				
Avg	267	355	433	<b>368</b>
Sd	33	20	34	<b>71</b>
Weighted Avg y ( $km$ )				
Avg	5,687	5,675	5,656	<b>5,670</b>
Sd	15	12	10	<b>17</b>

Table 3.5: Descriptive statistics of cluster results based on the existing distance measure described in section 3.3.2

applied to both cluster results to check whether the averages for each cluster are significantly different. The outcome of this effort is displayed in table 3.6. For the proposed spatio-temporal dissimilarity measure, only the differences between the clusters with respect to the values of eccentricity and area of SDE are statistically significant at the 0.05-level ( $P$ -values of 0.0028 and 0.0000 respectively), which indicates that the proposed spatio-temporal dissimilarity measure enables identifying groups of activity-travel sequences which do not cover the same geographical region, but which do share spatial characteristics. Conversely, for the clustering based on the existing dissimilarity measure, the table shows that, at the 0.05-level, statistically significant differences between the clusters are found for the weighted average  $(x, y)$ -coordinates ( $P$ -value of 0.0000), which proves that the existing dissimilarity measure is extremely suited to indicate geographical similarity while disregarding spatial proportions.

### 3.4.3 Design of Socio-Demographic Profiles

Starting from the clustering results obtained in the previous paragraph, socio-demographic profiles can be developed which characterize each cluster of similar activity-travel behaviour.

---

### 3.4. IDENTIFICATION OF GROUPS OF SIMILAR BEHAVIOUR

---

	Proposed measure	Existing measure
Area of SDE	0.0028*	0.1812
Eccentricity	0.0000*	0.3081
Weighted Avg x	0.6309	0.0000*
Weighted Avg y	0.2282	0.0000*

\* Significant on 0.05 level

Table 3.6: *P*-values of ANOVA tests

For this purpose, a decision tree is built based on the data because decision trees are particularly suited for this type of analysis. After all, decision tree algorithms are designed to process large datasets which include attributes of mixed data types (categorical or numerical) and to cope with missing values, unlike most statistical techniques pursuing the same goal (e.g. multinomial logit models) (Witten & Frank, 2000). Additionally, decision trees do not impose restrictions on the distribution of the data, as it provides a non-parametric or distribution-free algorithm (Breiman *et al.*, 1984). Most importantly, the outcome of a decision tree is easy to interpret and to convert into a set of if-then decision rules (Breiman *et al.*, 1984; Witten & Frank, 2000).

The decision tree algorithm applied in the current research is founded on the Classification and Regression Tree (CART) introduced by Breiman *et al.* (1984) and available in the *tree* toolkit included in the statistical software package *R*. The resulting classification tree is visualized in figure 3.27. Below each terminal node a bar plot reflects the segmentation of the predictor variable in that node. Figure 3.28 plots the standard deviation with respect to the predictor variable (in this case the cluster id) against the number of terminal nodes or leaves.

Table 3.7 displays more details concerning the decision tree and shows that the day of the week is the most distinguishing variable, especially with respect to cluster id 1, in which most sequences seem to be recorded on a working day (0=Monday, 1=Tuesday, ..., 6=Sunday). Next, the number of working hours per week (which is based on the theoretic work schedule, i.e. part time or full time, and not deduced from the activity sequences) appears to discriminate largely between cluster id 1 and cluster id 2. These findings are also supported in related researches where weekdays are distinguished from weekend days (Borgers *et al.*, 2002; Schlich, 2001; Wilson, 2001), and workers are treated differently from

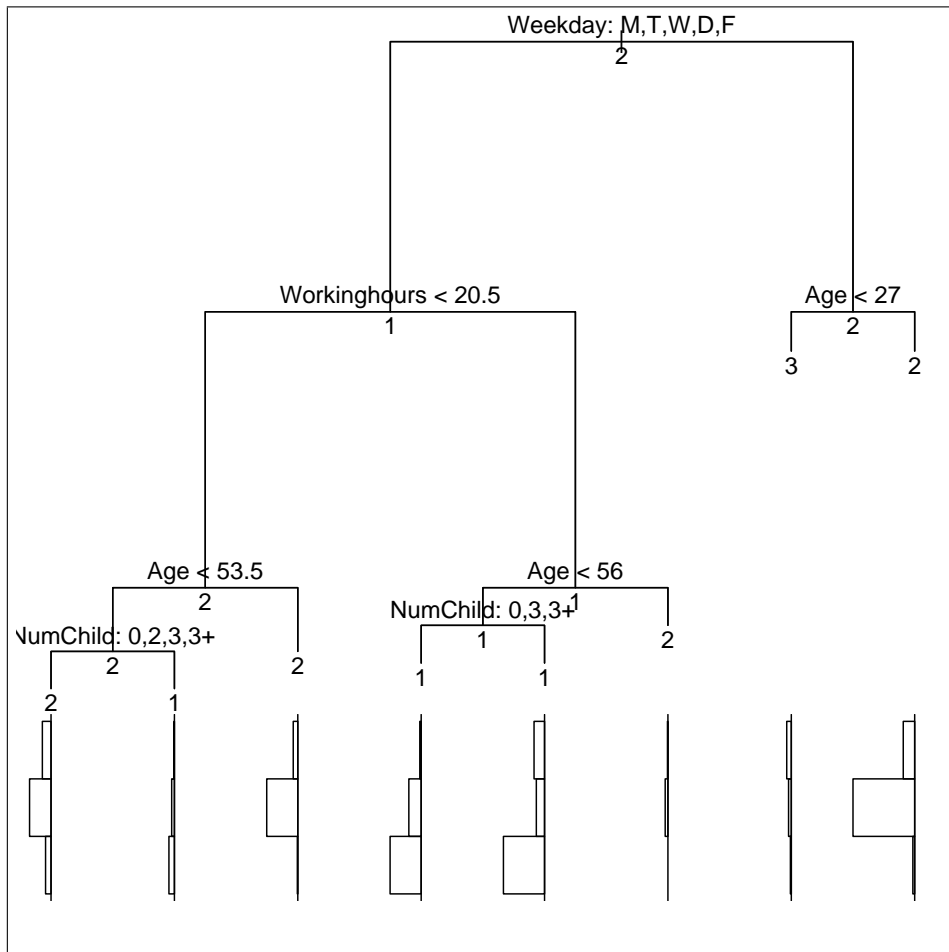


Figure 3.27: Decision tree attaching socio-demographical profiles to the cluster results (labels for weekday: *M* = Monday, *T* = Tuesday, *W* = Wednesday, *D* = Thursday, *F* = Friday, *S* = Saturday, *Z* = Sunday)

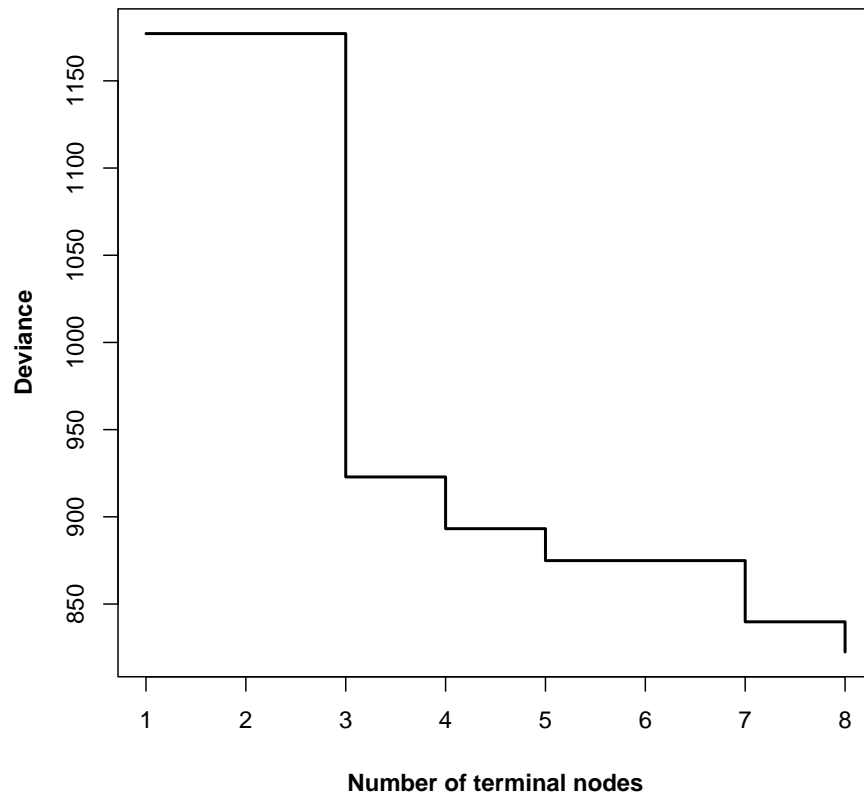


Figure 3.28: Overall standard deviation of predictor variable (i.e. cluster id)

non-workers (Anderson *et al.*, 2003; Bhat *et al.*, 2003; 2004; Borgers *et al.*, 2002; Bowman, 1998; Kulkarni & McNally, 2001; Schlich & Axhausen, 2004; Wilson, 2001).



Test rule	#cases	Pred.	%clu1	%clu2	%clu3	Leaf
Root	594	2	0.3182	0.5269	0.1549	
<i>Weekday : workday</i>	412	1	0.3081	0.2896	0.0960	
<i>Workinghours &lt; 20.5</i>	176	2	0.0421	0.2037	0.0505	
<i>Age &lt; 53.5</i>	97	2	0.0404	0.0892	0.0337	
<i>#child. : 0, 2, 3, 3+</i>	78	2	0.0202	0.0791	0.0320	*
<i>#child. : 1</i>	19	1	0.0202	0.0101	0.0017	*
<i>Age &gt; 53.5</i>	79	2	0.0017	0.1145	0.0168	*
<i>Workinghours &gt; 20.5</i>	236	1	0.2660	0.0859	0.0455	
<i>Age &lt; 56</i>	229	1	0.2660	0.0758	0.0438	
<i>#child. : 0, 3, 3+</i>	98	1	0.1145	0.0455	0.0050	*
<i>#child. : 1, 2</i>	131	1	0.1515	0.0303	0.0387	*
<i>Age &gt; 56</i>	7	2	0.0000	0.0101	0.0017	*
<i>Weekday : weekend</i>	182	2	0.0101	0.2374	0.0589	
<i>Age &lt; 27</i>	18	3	0.0034	0.0101	0.0168	*
<i>Age &gt; 27</i>	164	2	0.0067	0.2273	0.0421	*

Table 3.7: Outcome of decision tree attaching socio-demographical profiles to the cluster results

### 3.5 Conclusions

The current chapter provides an overview of the data required for the activity-based framework presented in this manuscript. To this end, the data effort is introduced and described by means of some general statistics. Nevertheless, because this data set contains activity-travel sequences collected by a number of individuals varying in their desires, needs, opportunities and constraints, these activity-travel patterns are assumed to be different in various aspects, e.g. the number of activity episodes contained in the sequences, their duration and geographical distribution.

Therefore this chapter designs a multidimensional measure to calculate the spatio-temporal (dis)similarity of activity-travel sequences, which incorporates the distances travelled between the activity locations and the relative location within a sequence rather than the distances between the locations of the sequences. After all, the methods advanced in literature generally are not able to compare sequences including non-nominal attributes and disregard the distances travelled within the separate sequences and the relative position of the locations within the same sequence with respect to one another. To this end, a multidimensional spatio-temporal dissimilarity measure is developed, which transforms and normalizes the geographical information contained by the  $(x, y)$ -coordinates of the activity locations into corresponding Angle/Arc Length (*AAL*)-trajectories to reflect the relative movements made within the sequence. Based on these *AAL*-trajectories, the traditional alignment technique can be used to calculate the mutual dissimilarities between the sequences.

Subsequently, a clustering technique, *pam* (Partitioning Around Medoids), is applied to the results of this dissimilarity approach in order to divide the data into a number of groups exhibiting a similar activity-travel pattern.

Finally, for the benefit of the scheduling framework, this chapter pursues a model which enables partitioning the individuals of the synthetic population based on a number of known attributes, such as age, gender, work status, work schedule and day of the week, into subgroups of individuals who are assumed to display similar activity-travel behaviour. As such, the current chapter concentrates on attaching socio-demographic profiles to each of the clusters determined previously by means of a classification tree. The main variables discriminating between the various types of activity-travel patterns within the observed data set

prove to be the day of the week and the work schedule.



## Chapter 4

# Design of the System

### 4.1 Introduction

To build a framework for simulating activity-travel behaviour, this chapter focuses on designing an agent-based micro simulation based on reinforcement learning. In the presented modelling framework, it is assumed that the population comprises a number of groups of individuals, who display similar activity-travel behaviour. As a result, the simulation of an activity-travel sequences of an individual from the synthetic population, can be based on the observed activity-travel data of all sequences which are supposed to match the intended individual most. Figure 4.1 visualizes the suggested modelling framework. The left side of this figure shows the input to the system, in particular activity-travel diaries, the corresponding diary attributes (e.g. day of the week) and socio-demographic data of the observed population. To start with, the activity, timing and location components in the diaries are processed in order to distinguish clusters of sequences displaying similar behaviour, as explained in section 3.4.1, based on the spatio-temporal dissimilarities between the activity-travel sequences computed according to the method introduced in section 3.3.3. For the purpose of assigning each individual in the synthetic population to the clusters defined here, the clusters are linked to the socio-demographic data and diary attributes to draw up a socio-demographic profile for each cluster, as explained in paragraph 3.4.3. In addition, the parameters for the reward functions incorporated in the reinforcement learning system, described below, are estimated for each cluster. Next, a so-called *prototype* agent is trained in the course of a

number of reinforcement learning episodes, based on this information. These components form the basis of the subsequent simulation steps.

On the right side of the figure, a synthetic population, representing the population which has to be simulated, is now inserted into the modelling framework. Each member of the synthetic population - also referred to as an agent - corresponds to an inhabitant of the selected area, and is first assigned to a cluster based on his socio-demographic profile. After that, all agents are initialized to the corresponding prototype agent and fine-tuned during an additional number of reinforcement learning episodes. The agents are now set and ready to generate the desired activity-travel patterns for the entire synthetic population.

The introduction of the prototype agent into the modelling framework requires some more explanation. The rationale underlying this agent is to incorporate general prior knowledge on states, actions and their corresponding rewards into each individual agent. After all, each prototype agent first gathers knowledge based on the reward functions, which are calibrated on the observed diary data matching the prototype agent, by running the reinforcement learning process a number of times. Afterwards, this knowledge is copied to all agents of the synthetic population which belong to the same cluster. As such, all agents in the synthetic population do not have to be trained from scratch, can proceed from the knowledge obtained from the corresponding prototype agent, and can refine this knowledge to their particular situation (e.g. geographical location, availability of transport modes, etc.). This way, the process of training the agents does not consume as much time as training each of these agents individually from scratch without any prior knowledge.

The actual reinforcement learning system is formulated according to the methodology described in chapter 2. The system is outlined in figure 4.2. The environment which the time of the system as well as the state of the network. Additionally, the agents interact with this environment. Their actions are controlled by a multi-actor reinforcement learning system incorporating batch/incremental regression tree function approximation and containing five decision modules.

The state of each agent is characterized by the activity in which he is currently engaged, its corresponding duration, location, travel mode and travel time, the travel mode of the current out-of-home tour, the next fixed activity and the time left to the earliest and last

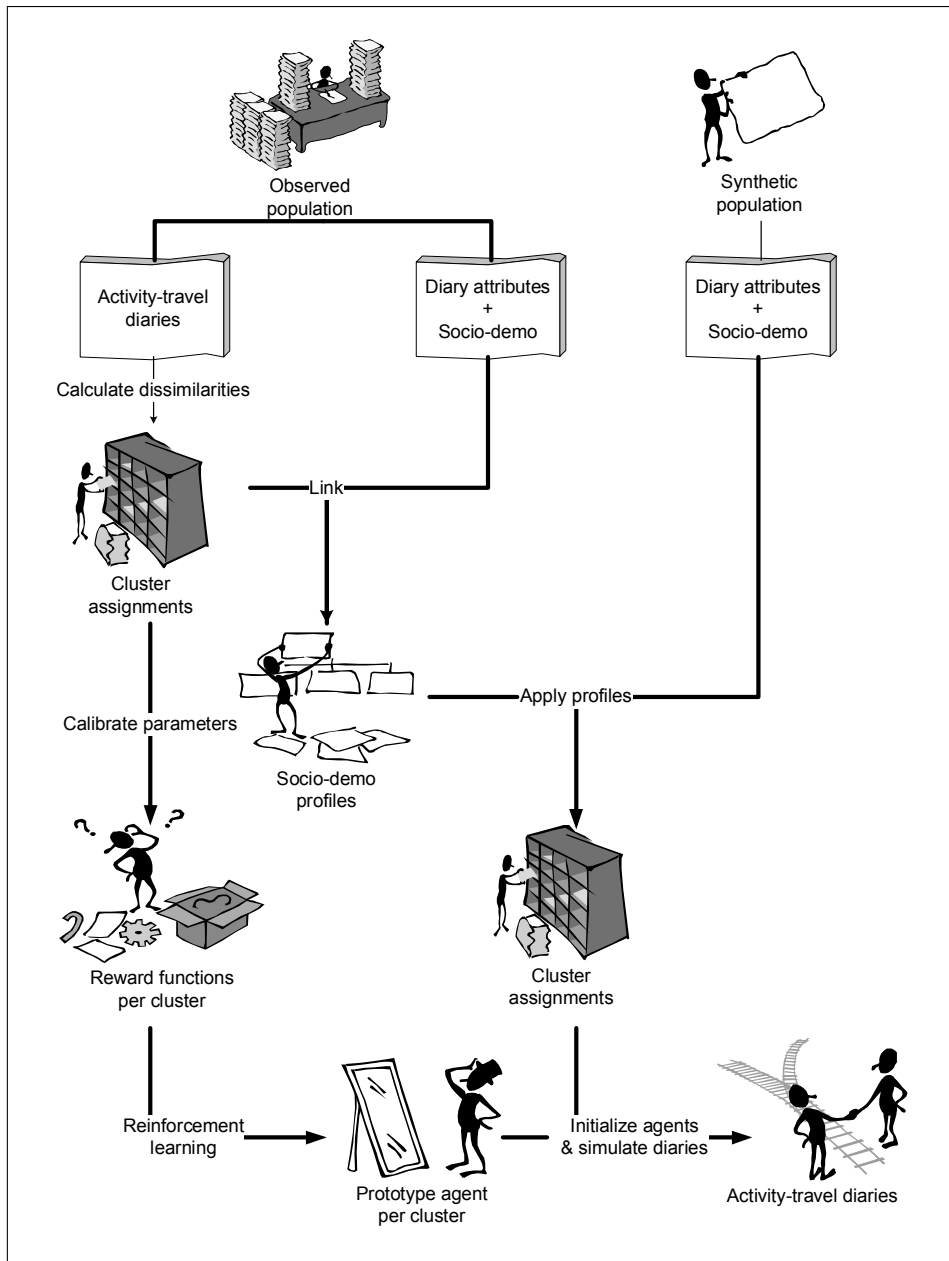


Figure 4.1: Agent-based micro simulation framework

starting time of this fixed activity (EST and LST respectively), the sequence of activities the agent has been executing since the midnight sleep activity, the total duration spent on each activity in the course of the day, (also since the midnight sleep activity), and the activity history (as measured by the time elapsed since the start of the last episode of each activity). Each agent's actions entail the activity type, duration, location and travel mode and are fixed in the course of the five reinforcement learning modules.

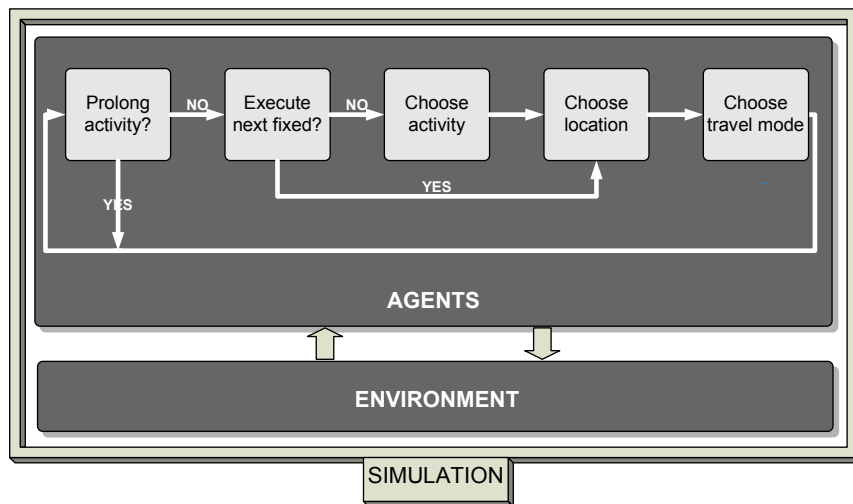


Figure 4.2: Reinforcement learning system

In the first module, the agent determines the activity duration by deciding if he wants to execute the current activity for one more time slot or if he wants to execute another activity. In case the agent does not want to prolong the duration of the current activity, the agent indicates in the second module whether he wants to execute the next fixed activity. If not, the agent selects another activity to perform in the third module. Each time a new fixed or flexible activity is selected, the fourth and fifth modules are called to select the activity location and the travel mode. Even though this decision process in itself is sequential, each decision module takes into account the impact of its decision on the optimal action of its successor by means of multi-actor reinforcement learning. Consequently, interactions between decision components as described in Gärling *et al.* (1997) and Joh *et al.* (2002), are accounted for. An example of the functioning of this scheduling process is presented in

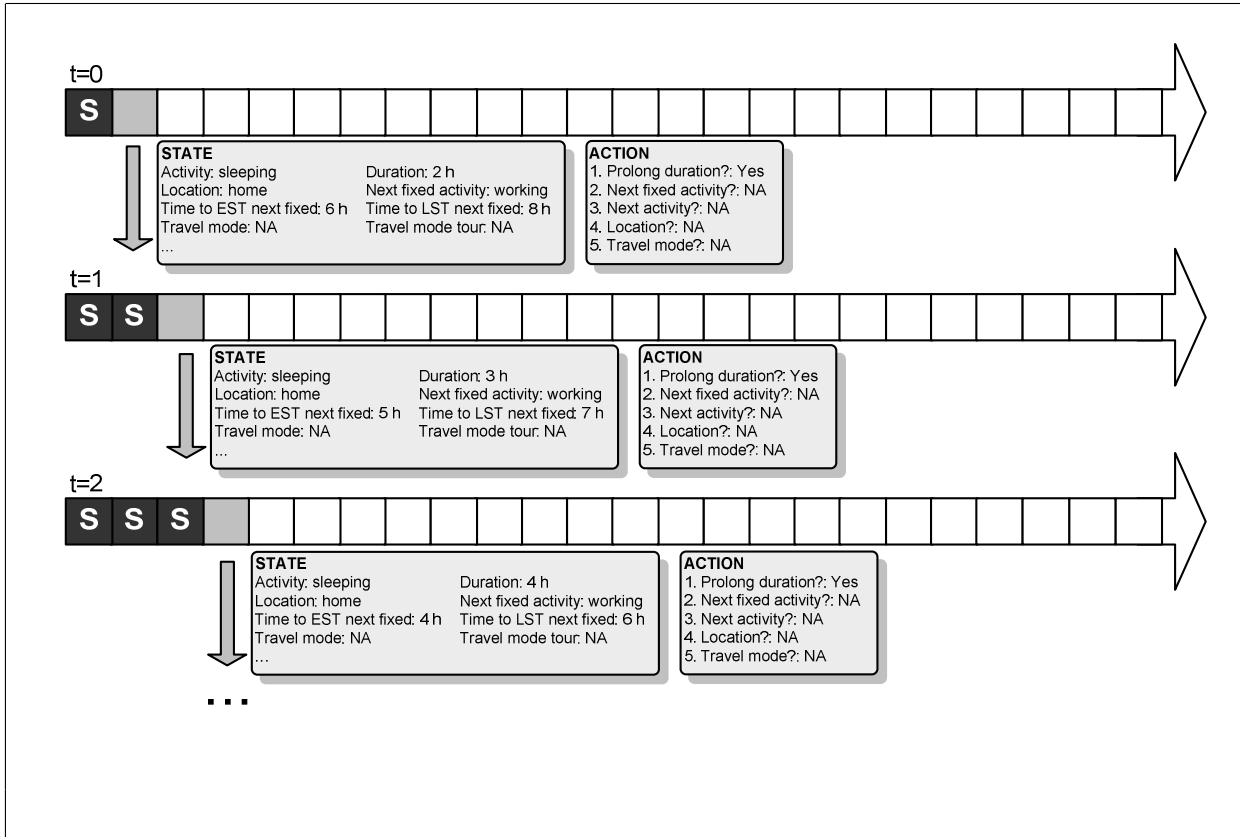


figure 4.3. A time step of one hour is assumed in this example.

The next sections elaborate on the specifications of each of the reinforcement learning modules. These sections draw special attention to the design of the reward functions for a number of reasons. Firstly, the reward functions can be derived either from frequencies in observed activity-travel diary data or based on stated-preference experiments (Janssens, 2005). Secondly, even though deterministic, reward functions can take on a number of functional forms, including a number of (independent) variables. Related research shows that the formulation of the reward function impacts the performance of reinforcement learning to a large extent, in particular in domains characterized by multiple goals, noisy states and inconsistent reinforcement (Mataric, 1994).

From this perspective, in order to enhance the learning process, Mataric (1994) describes some guidelines for formulating reward functions. First, in case an agent has to learn multiple objectives, a heterogeneous reinforcement function emerges; this signifies that each goal is reinforced individually. Furthermore, progress estimators - which record a metric of improvement with respect to a certain objective - can be used when attempting to reach a distant goal in a noisy world. After all, such a goal is traditionally characterized by a (strong) reward which is only assigned at the end of a sequence of actions when this ultimate goal is attained. In this case, progress estimators can also award the actions leading to this goal, reinforcing the intermediary actions required to reach the distant goal as well. The following sections elaborate on the design of the reward functions in each of the modules, bearing these requirements in mind.

Additionally, the functioning of each (isolated) module is illustrated. The results for each cluster defined in section 3.4, are discussed separately. To this end, the activity-travel sequences belonging to each cluster are split randomly into two groups: a training sample of 75% of the observed activity-travel diaries and a validation sample of 25% diaries. The training samples contain 142, 235 and 69 diaries and the validation samples include 47, 78 and 23 sequences in cluster one, two and three respectively. For each module the parameters for the reward functions are calibrated based on the activity-travel data contained in the training sample. These reward functions are then applied to initialize the three prototype agents, which - on their turn - found the basis of test agents, which correspond to an activity-



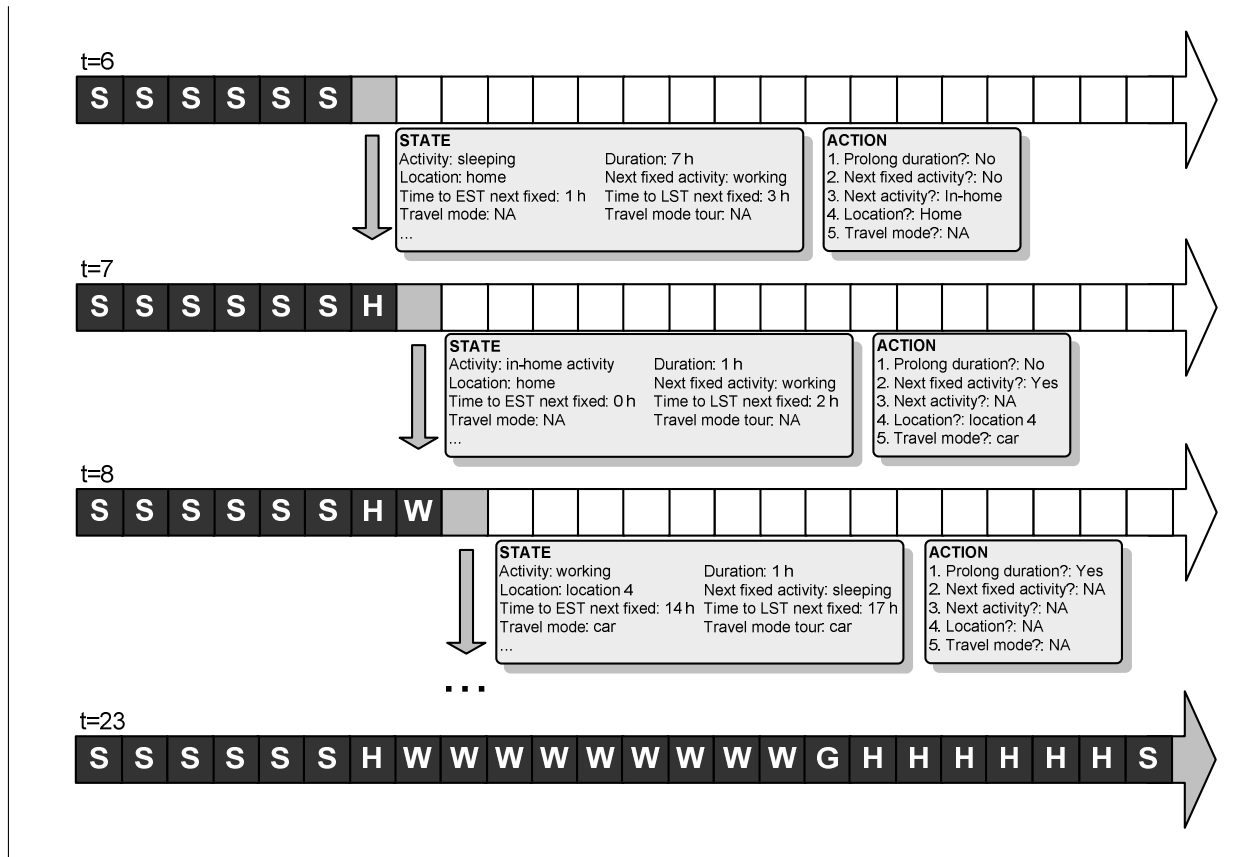


Figure 4.3: An example of the scheduling process illustrated

travel sequence of the validation set. Next, the test agents are fine-tuned and used to generate the desired activity-travel output. Finally, to evaluate the performance of each module, the output is compared to these validation sequences.

## 4.2 Module 1: Duration

### 4.2.1 Reward Function

The first module concentrates on determining the duration of the selected activity. To this end, at each time step the agent chooses to prolong the activity duration with one time step (i.e. stay) or to proceed to the next activity (i.e. move), as inspired on work introduced by Charypar *et al.* (2004). This module is a typical example of a learning process pursuing a distant goal.

A traditional reward function guiding this decision module would assign a reward only when the agent decides to move to the next activity. The magnitude of this reward would depend on the final duration of the activity. Figure 4.4 plots an example of such a reward function for the working activity. The reward function for executing the activity for one more time slot equals zero, whereas the reward function for moving to the next activity is based on the observed cumulative density distribution with respect to the duration for this activity type. The grey line reflects a curve fitted based on the observed training data (i.e. the grey dots).

However, the performance of this decision module can be improved by reformulating the reward function more carefully employing progress estimators. Consequently, the reward function indicates to which amount the agent's action contributes to the final outcome. Actions leading to the desired goal are rewarded more than those resulting in an unfavourable state. As such, the reward function for the duration of an activity can be inspired on the concept of decreasing marginal utility. For example, in case of a working activity, prolonging the activity for one more time step (e.g. 5 minutes) pays off more than moving to the next activity when the current duration when the current duration equals 10 minutes, while this reward is less if the duration of the current working activity already equals 480 minutes.

An example of such reward function is shown in figure 4.5. The reward function for the

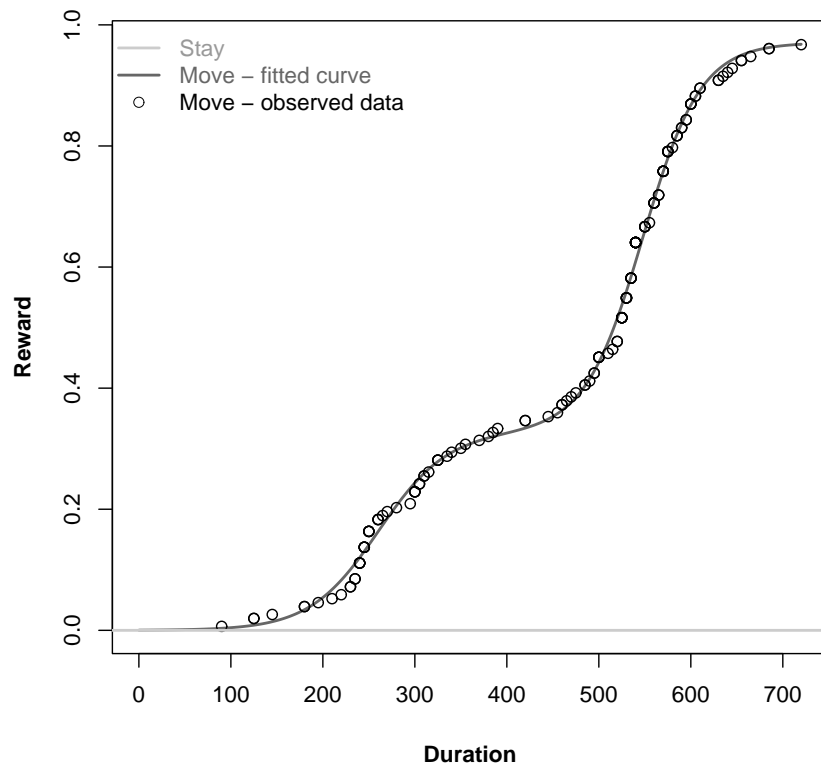


Figure 4.4: Traditional reward function in module 1 for the working activity of cluster 1

action “move” is identical to the traditional reward function, but the reward function for the action “stay” equals one minus the reward function for the action “move”. That way, the agent receives immediate feedback for both actions, and not only for the action move. As prolonging the current activity now also pays off immediately, the agent is stimulated to prolong the current activity. The difference between these two reward schemes is illustrated in figure 4.6. In this figure, the agent prolongs the duration of the current activity up to 480 minutes. Thereafter, he decides to move to the next activity, for which the reward is equal in both reward schemes.

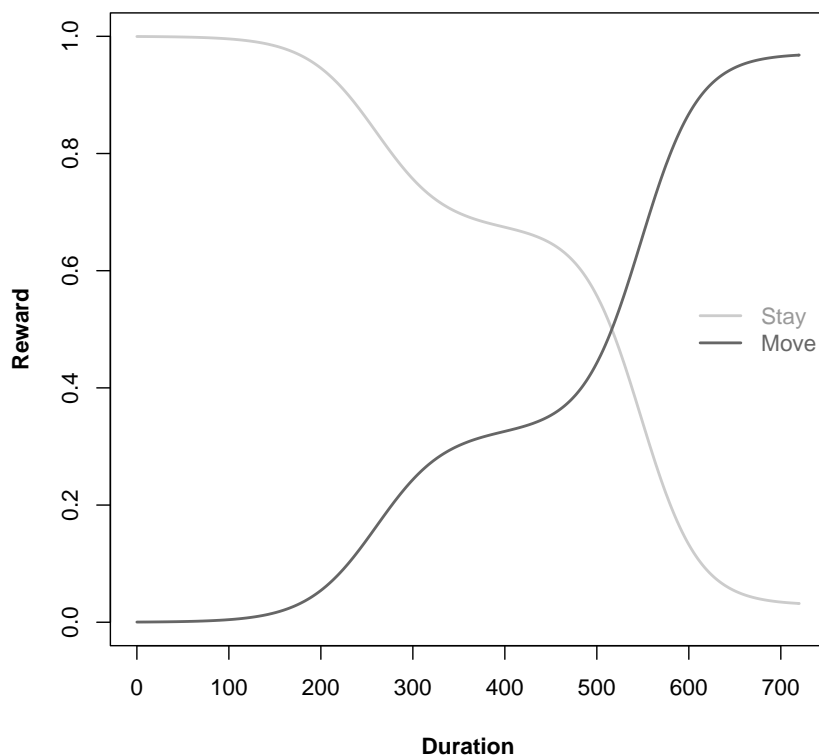


Figure 4.5: Reward function using progress estimators in module 1 for the working activity of cluster 1

Obviously, other reward functions are feasible as well. Figure 4.7 plots an alternative

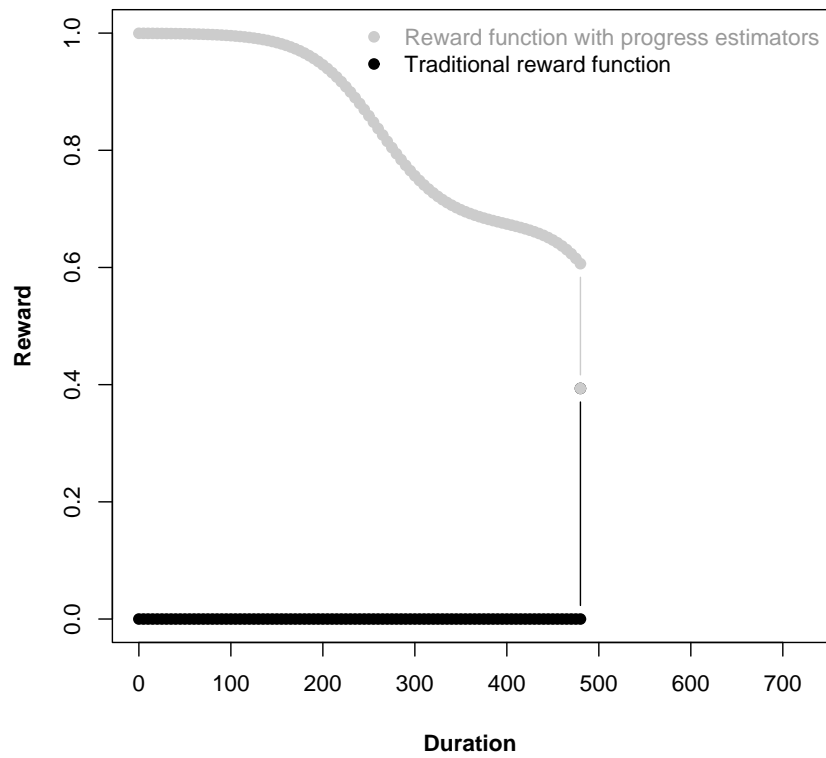


Figure 4.6: Example of rewards assigned in module 1 for an agent prolonging the duration of the current working activity up to a duration of 480 minutes and then moving to a next activity

reward function for the working activity. The discontinuity in this reward function is based on the minimum, maximum and average duration of this activity. In this case, the agent receives a rather high, slightly decreasing reward for prolonging the activity duration as long as the duration is below the minimum observed duration. Between the minimum and the average observed duration, the reward is reduced, but is still positive. When the duration is situated in the interval between the average and the maximum observed duration, a small penalty is assigned to discourage the agent to prolong the duration of this activity. After that, the reward of prolonging the activity duration drops considerably. On the other hand, the agent receives a penalty for moving to the next activity while the activity duration is smaller than the average observed duration. When the duration exceeds the average observed duration, the reward for moving equals zero.

Additionally, this alternative reward function takes into account the total duration spent on each activity in the course of the day. If the total duration of a certain activity surpasses the maximum total duration observed in the data, the agent is assigned a rather large negative reward.

Both reward schemes are tested in the present research. For the first alternative, the cumulative distribution function is used to fit the reward function. In this respect, two functional forms - which are assumed to reflect the utility of executing a certain activity for a certain duration - are fitted to the data. The functional form representing the data best, based on the residual standard error, is selected for this experiment. The first function is the logistic model, as proposed in Joh *et al.* (2004). In the current research, the general form of a utility function matching this model, is assumed to be a symmetric *S*-shaped curve:

$$U = U_{min} + \frac{U_{max}}{1 + e^{\beta(\alpha-d)}}. \quad (4.1)$$

Here  $d$  is the activity duration,  $U$  is the utility,  $U_{min}$  is the minimum utility and  $U_{max}$  is the maximum attainable utility for the activity. In their research, Joh *et al.* (2004) assume that  $U_{max}$  is a function of the activity history of the activity, determining as such the desirability or need to conduct this activity. However, to extract the reward solely caused by the duration in this module, the parameter  $U_{max}$  does not depend on the activity history. Yet, this assumption is applied when selecting which activity to perform in module three (cf.



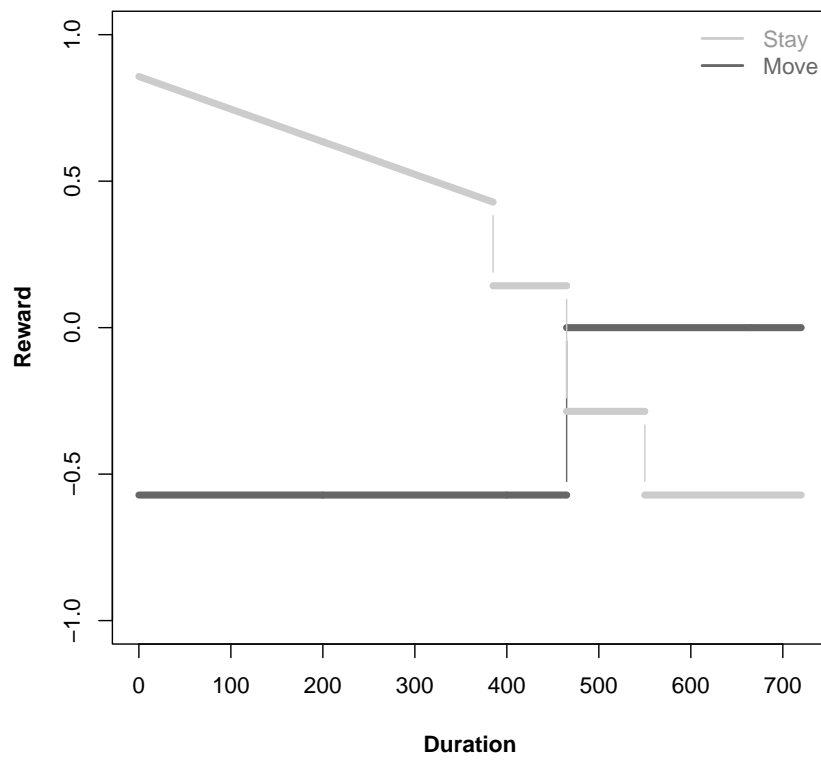


Figure 4.7: Alternative reward function using progress estimators in module 1 for the working activity of cluster 1

section 4.4). The parameter  $\beta$  determines the slope of the curve and the parameter  $\alpha$  the inflection point. The second functional form is the double logistic model, which is merely a variant of the first one. The relationship between the utility and the duration for an activity can be expressed as:

$$U = \frac{U_{max}^{(1)}}{1 + e^{\beta^{(1)}(\alpha^{(1)} - d)}} + \frac{U_{max}^{(2)}}{1 + e^{\beta^{(2)}(\alpha^{(2)} - d)}}. \quad (4.2)$$

This function looks like a double  $S$ -shaped curve. The parameters  $U_{max}^{(1)}$ ,  $\beta^{(1)}$  and  $\alpha^{(1)}$  correspond to the lower  $S$  of the curve while the parameters  $U_{max}^{(2)}$ ,  $\beta^{(2)}$  and  $\alpha^{(2)}$  refer to the upper  $S$  of the curve. To illustrate the tuning of these parameters, table 4.1 displays for cluster 1 the parameters of the best fit for each activity type for which the number of observed instances exceeds 20. These utility functions form the basis of the reward functions and are therefore reformulated to represent progress estimators as indicated above.

Table 4.2 records for each feasible activity type the bounds of the intervals used for the alternative reward function. The parameter settings are calibrated on the basis of the data of the 142 training sequences in cluster 1. The information contained in the validation set is thus not included in these parameters.

Activity type	Functional form	Parameters	Plot
In-home activities	Double logistic	$\alpha^{(1)} = 52.1767$ $\beta^{(1)} = 0.0711$ $U_{max}^{(1)} = 0.4487$ $\alpha^{(2)} = 315.7392$ $\beta^{(2)} = 0.0126$ $U_{max}^{(2)} = 0.5482$	
Working	Double logistic	$\alpha^{(1)} = 260.4623$ $\beta^{(1)} = 0.0268$ $U_{max}^{(1)} = 0.3277$ $\alpha^{(2)} = 547.8893$ $\beta^{(2)} = 0.0317$ $U_{max}^{(2)} = 0.6431$	

Continued on Next Page...

Activity type	Functional form	Parameters	Plot
Bring/Get	Logistic	$\alpha = 20.6662$ $\beta = 0.1485$ $U_{min} = 0.0000$ $U_{max} = 0.9290$	
Sleeping	Logistic	$\alpha = 440.9003$ $\beta = 1.2002$ $U_{min} = 0.0000$ $U_{max} = 0.8874$	

Table 4.1: Parameters of the best curve fitted to the observed data of cluster 1

---

<b>Activity type</b>	<b>Avg.dur.</b>	<b>Min.dur.</b>	<b>Max.dur.</b>	<b>Max.tot.dur.</b>
In-home activities	191	19	401	585
Working	164	385	549	639
Services	29	19	76	76
Out-of-home eating	66	0	129	80
Grocery shopping	26	5	58	75
Non-daily shopping	89	0	165	132
Education	139	167	446	528
Social activity	204	81	489	543
Leisure	102	70	274	283
Bring/Get	41	0	77	83
Touring	0	0	0	0
Other	0	0	0	0
Sleeping	67	405	473	967

---

Table 4.2: Parameters of the alternative reward functions based in module 1 on the observed data of cluster 1

#### 4.2.2 Validation

As this evaluation is intended to show the added value of the duration module only, the system assigns a duration to each activity of a given sequence of activities, which is not determined by the system. To this end, the activity sequences of the validation cases serve as input to the experiment. The system thus generates 148 agents - each corresponding to one of the test agents - attaching a duration to each activity of these predetermined sequences. Two experiments are conducted: one experiment utilizes the alternative reward functions, whereas the other one applies the reward functions based on the curves estimated on the observed data. In the latter case, when an activity type is observed less than 20 times in the sequences included in the training data, the alternative reward function is used as well.

Tables 4.3 to 4.5 summarize the results of these experiments for clusters one to three respectively, by means of the average and standard deviation of the durations predicted for each of the validation sequences, and compare these statistics to those observed in the training set and the validation set. Furthermore, with respect to the observed training and validation data, it should be mentioned that some activities exhibit a large standard deviation (e.g. the social activity). This is due either to the fact that only a low number of episodes of these activities are observed in the data, or to a rather large variability of attributes - such as duration - in the activity categories. The latter issue is also remarked by

Doherty (2006). In particular, a large variability of the duration is observed for the in-home activities, as this activity actually fills the gaps between various out-of-home activities.

Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	283	210	191	100	203	189	170	25	296	92
Working	153	467	164	55	438	160	519	2	444	113
Services	4	48	29	1	10	-	55	-	65	-
Out-of-home eating	19	63	66	13	59	50	67	6	64	2
Grocery shopping	11	32	26	2	55	7	53	4	43	4
Non-daily shopping	9	76	89	2	198	272	75	0	85	7
Education	13	307	139	3	193	61	312	3	313	3
Social activity	15	285	204	4	151	83	275	0	285	0
Leisure	13	172	102	2	95	7	163	25	175	7
Bring/Get	29	36	41	8	31	43	26	2	36	12
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	284	439	67	94	442	71	450	1	444	2

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.3: Validation results for the duration module for cluster 1

Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	403	505	379	123	525	361	435	0	500	0
Working	11	237	134	4	229	220	240	0	239	3
Services	20	115	105	2	85	49	115	0	115	0
Out-of-home eating	14	81	45	7	112	114	93	17	85	0
Grocery shopping	29	36	26	8	41	35	41	18	43	6
Non-daily shopping	22	65	46	5	44	47	64	9	70	0
Education	9	253	107	5	331	186	253	3	255	0
Social activity	33	141	98	16	119	61	105	0	143	8
Leisure	29	183	139	5	180	127	128	32	201	11
Bring/Get	53	30	33	12	16	7	23	17	33	8
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	474	456	94	157	467	100	447	12	463	2

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.4: Validation results for the duration module for cluster 2



Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	66	409	304	21	385	289	288	16	430	67
Working	2	373	4	2	320	262	370	0	370	0
Services	0	0	0	0	-	-	-	-	-	-
Out-of-home eating	2	210	127	1	40	-	210	-	205	-
Grocery shopping	0	0	0	1	25	-	5	-	5	-
Non-daily shopping	1	90	0	0	-	-	-	-	-	-
Education	1	500	0	0	-	-	-	-	-	-
Social activity	5	562	133	6	341	97	563	3	576	2
Leisure	4	336	261	4	150	94	349	5	464	178
Bring/Get	3	13	8	1	100	-	5	-	20	-
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	138	449	46	46	440	88	211	15	450	0

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.5: Validation results for the duration module for cluster 3

These statistics already lift the veil on the predictive capability of the current module. However, to quantify this, table 4.6 and 4.7 display the root of the average of the squared difference between the predicted duration and the average of the observed duration of the corresponding activity calculated based on all cases in the training dataset on the one hand (i.e.  $\sqrt{(\hat{d} - \bar{d}_{training\ set})^2/n}$ ), and the same difference between the predicted duration and the corresponding activity of the sequence matching the predicted sequences in the validation dataset on the other hand (i.e.  $\sqrt{(\hat{d} - \bar{d}_{validation\ set})^2/n}$ ). The former difference reflects the deviation of the predicted duration with respect to the average of the observed duration of the training dataset, while the latter reveals the deviation of the predicted duration from the corresponding observed duration in the validation dataset. The columns entitled “Obs” display the deviance of the observed duration in the validation set from the baseline model, assigning the average duration of the activity type in the training data to each activity episode in the observed validation sequences.

A brief comparison of both tables shows that the predicted durations match the average duration observed in the training set (as can be read from table 4.6) better than the average duration of the observed validation sequences (as displayed in table 4.7), which is as expected as the models are fitted based on the training data. Furthermore, table 4.7 indicates that in the majority of the cases the magnitude of the deviance of the predicted durations with respect to the observed duration in the validation data falls within the same order of magnitude as the prediction error of the baseline model. From these small-scale experiments can thus be concluded that this module - regardless of the type of reward function - is able to assign the duration of an activity episode well.

Activity type	Cluster 1			Cluster 2			Cluster 3		
	Obs	Sim (1)	Sim (2)	Obs	Sim (1)	Sim (2)	Obs	Sim (1)	Sim (2)
In-home activities	188	47	126	360	70	5	283	122	68
Working	161	52	114	191	3	3	192	3	3
Services	38	8	18	46	0	0	-	-	-
Out-of-home eating	48	7	2	110	20	4	170	0	5
Grocery shopping	24	21	11	33	18	9	25	5	5
Non-daily shopping	228	1	10	47	8	5	-	-	-
Education	124	6	7	183	2	2	-	-	-
Social activity	152	10	0	63	36	8	238	3	14
Leisure	77	20	6	113	62	20	203	13	200
Bring/Get	41	10	12	15	17	9	87	8	7
Touring	-	-	-	-	-	-	-	-	-
Other	-	-	-	-	-	-	-	-	-
Sleeping	71	11	6	100	15	7	88	239	1

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.6: Root of average squared difference between the predicted duration and the average observed duration for the corresponding activity calculated in the training dataset

Activity type	Cluster 1			Cluster 2			Cluster 3		
	Obs	Sim (1)	Sim (2)	Obs	Sim (1)	Sim (2)	Obs	Sim (1)	Sim (2)
In-home activities	188	185	199	360	370	360	283	304	300
Working	161	178	192	191	191	191	192	192	192
Services	38	45	55	46	46	46	-	-	-
Out-of-home eating	48	49	47	110	103	109	170	170	165
Grocery shopping	24	4	15	33	44	35	25	20	20
Non-daily shopping	228	228	227	47	48	50	-	-	-
Education	124	129	130	183	184	183	-	-	-
Social activity	152	143	152	63	61	64	238	240	251
Leisure	77	69	81	113	114	112	203	216	343
Bring/Get	41	39	40	15	18	20	87	95	80
Touring	-	-	-	-	-	-	-	-	-
Other	-	-	-	-	-	-	-	-	-
Sleeping	71	71	71	100	102	100	88	245	88

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.7: Root of average squared difference between the predicted duration and observed duration of the activity in the corresponding validation sequence

Though, the tables show that the outcomes for both reward schemes are comparable, the alternative reward function is utilized in this project, under the supposition that this reward scheme generates better results in the presence of noise. After all, when introducing the remaining modules of the suggested multi-actor reinforcement learning framework, the rewards obtained in the current module are biased by the rewards received as a result of actions taken in the subsequent modules.

Bearing this in mind and given that the alternative reward function includes discontinuous, discrete rewards and penalties - as opposed to the continuous reward functions -, the alternative reward function is assumed to enable the duration module to draw up a more robust regression tree. After all, clear-cut borders within the reward scheme are easier to distinguish by a decision tree algorithm compared to a smoothly changing reward function, because a decision tree searches for significant differences in the dependent variable in order to partition the attribute space.

Finally, the added value of the regression tree function approximation incorporated in the reinforcement learning agent can be illustrated by conducting this experiment for traditional reinforcement learning agents as well. The expected values of the actions taken by these agents are stored in a so-called  $Q$ -table. To enable distinguishing action values dependent on the state of the system, the agents enhanced with regression tree function approximation are able to include a large number of state variables in their decision process; for instance in this module, an agent takes into account the time of the day, the current activity, the duration of the current activity episode, the time spent on the current activity throughout the entire day, the location of this activity, the travel mode used to get to this location and the travel time required, the travel mode of the current home-based tour, the next fixed activity and the time remaining till the earliest and last starting time of this next fixed activity.

Yet, due to limited memory capacity, the  $Q$ -table for the present module cannot incorporate all of these state variables and thus only takes into consideration the time of the day, the activity which is being performed and its duration. Consequently, even in this simplified setting the  $Q$ -table contains 1,078,272 cell entries if the time slot equals 5 minutes: 288 time slots in a day  $\times$  13 activity types  $\times$  144 values for the activity duration (which is

restricted to 720 minutes and which increments by the selected time slot)  $\times$  2 action values (either continue executing the current activity or move to the next activity). Obviously this  $Q$ -table still requires a large amount of memory capacity. Furthermore, as already indicated in section 2.2.4 each entry of this table should preferably be visited a number of times in order to be able to estimate its value accurately. The outcome generated by a traditional reinforcement learning system is presented in tables 4.8 to 4.10.

Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	283	210	191	100	203	189	143	19	199	36
Working	153	467	164	55	438	160	458	53	442	48
Services	4	48	29	1	10	-	70	-	65	-
Out-of-home eating	19	63	66	13	59	50	77	7	71	7
Grocery shopping	11	32	26	2	55	7	55	0	55	0
Non-daily shopping	9	76	89	2	198	272	85	7	85	0
Education	13	307	139	3	193	61	222	58	207	12
Social activity	15	285	204	4	151	83	155	0	138	5
Leisure	13	172	102	2	95	7	145	0	115	0
Bring/Get	29	36	41	8	31	43	45	0	56	7
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	284	439	67	94	442	71	376	135	332	156

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.8: Validation results for the duration module for cluster 1 based on traditional reinforcement learning agents

Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	403	505	379	123	525	361	404	96	465	118
Working	11	237	134	4	229	220	240	0	213	55
Services	20	115	105	2	85	49	115	0	115	0
Out-of-home eating	14	81	45	7	112	114	85	0	89	2
Grocery shopping	29	36	26	8	41	35	63	8	60	0
Non-daily shopping	22	65	46	5	44	47	77	13	84	8
Education	9	253	107	5	331	186	229	25	198	55
Social activity	33	141	98	16	119	61	94	19	123	38
Leisure	29	183	139	5	180	127	116	2	178	16
Bring/Get	53	30	33	12	16	7	48	18	46	10
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	474	456	94	157	467	100	398	113	399	143

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.9: Validation results for the duration module for cluster 2 based on traditional reinforcement learning agents



Activity type	Training			Validation			Simulated(1)		Simulated(2)	
	#	Avg	Sd	#	Avg	Sd	Avg	Sd	Avg	Sd
In-home activities	66	409	304	21	385	289	302	22	410	0
Working	2	373	4	2	320	262	370	0	370	0
Services	0	0	0	0	-	-	-	-	-	-
Out-of-home eating	2	210	127	1	40	-	210	-	210	-
Grocery shopping	0	0	0	1	25	-	5	-	5	-
Non-daily shopping	1	90	0	0	-	-	-	-	-	-
Education	1	500	0	0	-	-	-	-	-	-
Social activity	5	562	133	6	341	97	443	188	442	191
Leisure	4	336	261	4	150	94	296	88	294	93
Bring/Get	3	13	8	1	100	-	15	-	20	-
Touring	0	0	0	0	-	-	-	-	-	-
Other	0	0	0	0	-	-	-	-	-	-
Sleeping	138	449	46	46	440	88	196	30	409	84

(1) Simulated validation set generated by means of curves-based reward functions

(2) Simulated validation set generated by means of alternative reward functions

Table 4.10: Validation results for the duration module for cluster 3 based on traditional reinforcement learning agents

This experimental setting requires at least 25,000 learning episodes to approach the results produced at the end of 5,000 learning episodes by the reinforcement learning agents enhanced with regression tree function approximation. However, these results could be even more improved by adding more learning episodes. This proposition ensues from the observation that mainly the duration of activities lasting for a long time (e.g. in-home activities, sleeping and education) are systematically underestimated. This is caused by the fact that a series of optimal actions - i.e. continue executing the current activity - is required before being able to reach a state including a long duration. Upon arrival in such state, which the agent has not visited before, the agent does not favour any of the two feasible actions and selects one action randomly and only after the same series of optimal actions occurs, the agents can reach this state again to explore the other action. The optimal duration can thus only be extended by one more time slot after this series of optimal actions is learnt one time slot at a time. In case of the reinforcement learning agents enhanced with regression tree function approximation, the regression tree enables generalizing for instance over the activity duration, and, as such, learning this series of optimal actions for a number of time slots at once.

Furthermore, as part of these experiments, special attention is paid to the definition of the exploration rate, which is introduced in section 2.2.2. While the exploration rate in the enhanced agents is derived from the regression tree, the exploration rate in the traditional agents is determined so as to advance this succession of optimal actions. To this end, the exploration rate has to decrease rapidly, allowing the agent sooner to select the best action rather than a random action. The exploration rate implemented in the traditional agents is visualized in figure 4.8 can be expressed as follows:

$$p_{explore} = 1 - \frac{\ln(\text{episode number})}{\ln(25,000)}. \quad (4.3)$$

The exploration rate for the agents enhanced with regression tree function approximation is based on the reduction of the variance as measured by the proportion of the average variance of the leaves, weighted according to the number of cases assigned to each leaf, with respect to the root variance of the regression tree. The formula for calculating this exploration rate is shown here:

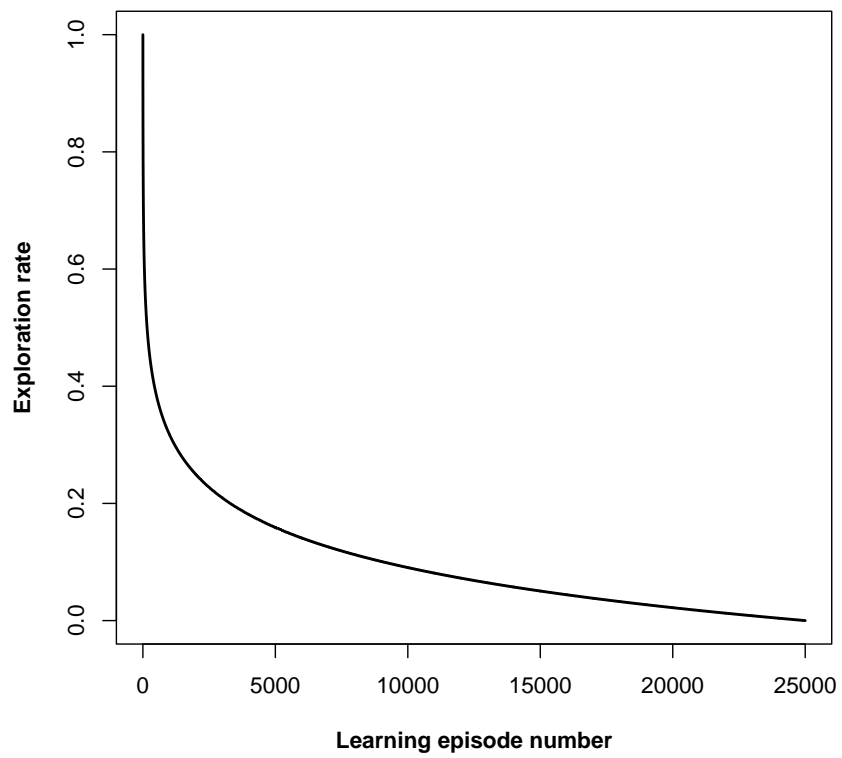


Figure 4.8: Exploration rate implemented in module 1 for the traditional reinforcement learning agents

$$p_{explore} = 5 * \frac{\sum (\sigma_{leaf}^2 * n_{leaf})}{\sum n_{leaf} - 1} \cdot \frac{1}{\sigma_{root}^2}. \quad (4.4)$$

In the course of testing the algorithm, it turned out that the relative reduction in the root variance has to be multiplied by 5 to obtain a meaningful exploration rate. It should be noted though that the nature of the exploration rate forces the outcome of equation 4.4 to be clipped within the interval  $[0, 1]$ . The values of the exploration rates as generated by the learning algorithm are displayed in figure 4.9. These plots also reveal that the regression tree enables to substantially reduce the variance in all the collected cases by splitting the dataset.

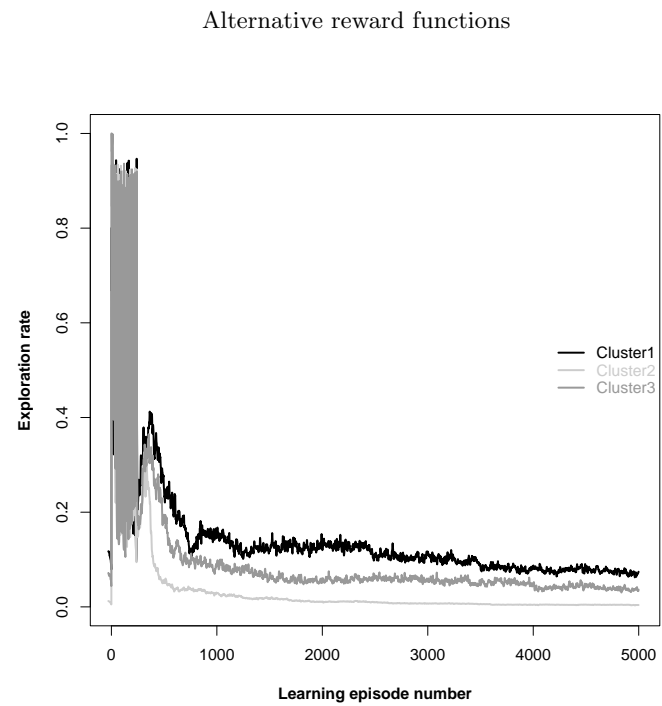
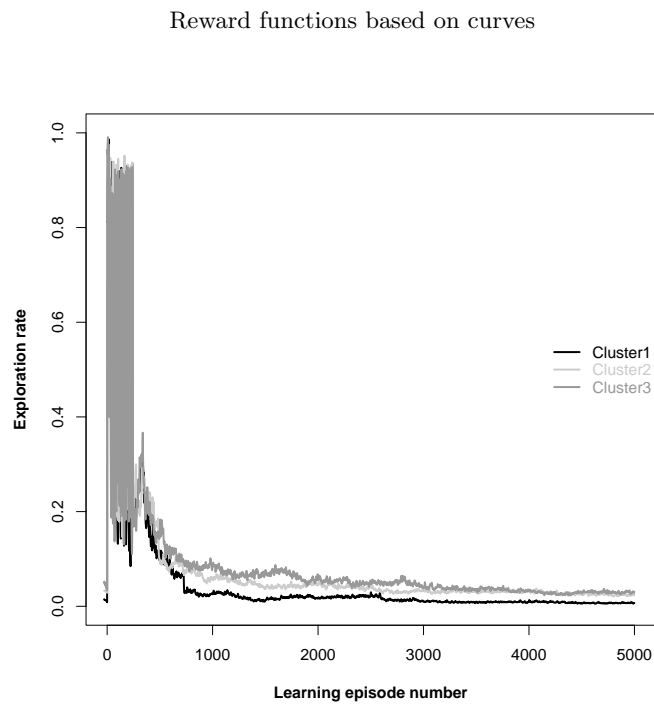


Figure 4.9: Exploration rate generated in the course of the learning process for module 1 for the reinforcement learning agents enhanced with regression tree function approximation

## 4.3 Module 2: Fixed Activities

### 4.3.1 Reward Function

In case the agent does not prolong the duration of the current activity, the agent decides in the second decision module whether to execute the next fixed activity or not. This decision module supports the notion of the existence of an activity skeleton containing some activities, such as working and sleeping, which tend to be rather fixed in time and space (supported by among others Arentze & Timmermans (2005a), Chapin (1974), Ettema & Timmermans (1997b) and Jones (1979)). The reward attached to this decision depends on the time of the day: if the agent decides to execute the next fixed activity, he only receives a positive reward if he starts this fixed activity between its (given) earliest and last starting time (i.e. EST and LST respectively). Outside these starting time boundaries, the agent receives a penalty (i.e. negative reward) for wanting to execute the fixed activity. Furthermore, between these starting time boundaries, a penalty is assigned as long as the agent does not execute the fixed activity. Due to the incorporation of multi-actor reinforcement learning, this penalty is also accounted for in the previous module. The reward function of module 2 is displayed in figure 4.10.

In the current research, only two activity types are assumed to be fixed: the working and evening sleep activity. The starting time boundaries of these fixed activities are deduced from their observed starting times. The earliest starting time corresponds to the 25th percentile and the last starting time equals the 75th percentile. For clusters 2 and 3 the working activity is only observed in respectively 5% and 3% of the sequences, whereas it is observed at least once in 94% of the sequences in cluster 1. Therefore, in the former clusters the working activity is not treated as a fixed activity. The outcome of this module is presented in the next section, describing module 3, as these two modules concurrently define the activity to be executed.

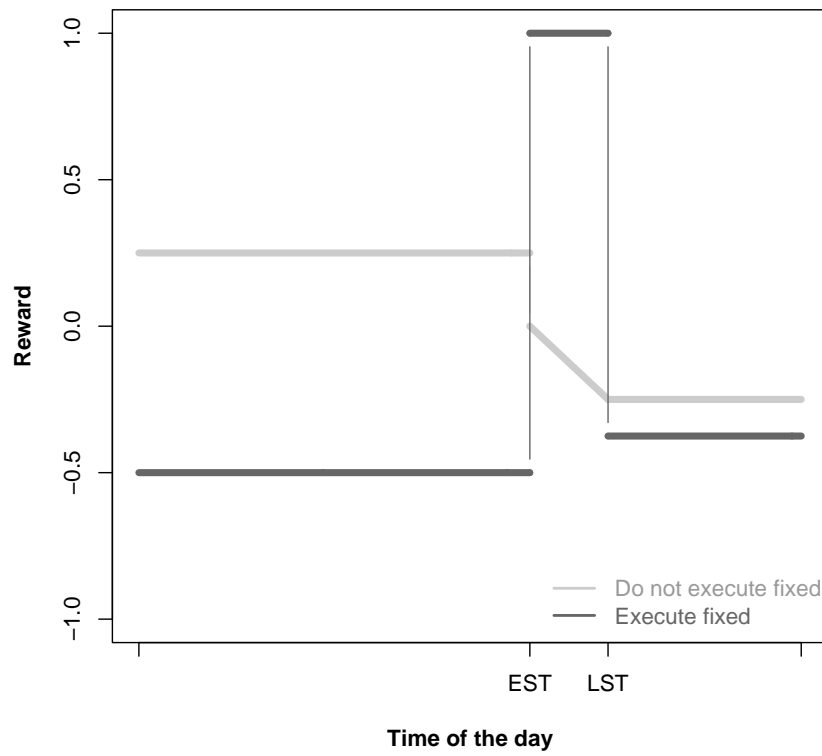


Figure 4.10: Reward function for module 2 for the fixed working activity

## 4.4 Module 3: Activity selection

### 4.4.1 Reward Function

The third module focuses on choosing the next activity to be executed in case the agent does not want to perform the next fixed activity. Two reward schemes guiding this selection process are tested. The first reward scheme relies on the activity pattern executed since midnight, whereas the second accounts for the history of the activities in the sequence. The former reward consists of a progress estimator deduced from the sequence composed in the course of the scheduling process, and is calculated by means of a sequence alignment method (SAM) (Joh *et al.*, 2001). The reward of each action reflects the improvement in matching the (temporary) sequence to the observed input sequences:

$$R^t = \frac{\sum_j [w_j (SAM_j^t - SAM_j^{t-1})]}{\sum_j w_j}. \quad (4.5)$$

The reward is based on the weighted difference of unidimensional distance measures calculated by means of a sequence alignment method with respect to the observed sequences.  $SAM_j^{t-1}$  represents the dissimilarity between the existing sequence of activities and the observed sequence  $j$ , whereas  $SAM_j^t$  is calculated between this sequence appended with the selected activity and the observed sequence  $j$ .  $w_j$  assigns a weight according to the relative importance of simulating an activity pattern that matches the observed sequence  $j$ . The sequence alignment method used is founded on the method described in Joh *et al.* (2001) and Wilson (1998a), and summarized in section 3.3.1. For this purpose, the elementary operations - i.e. insert, delete or substitute - of this alignment method can be given different weights.

The alternative reward function underlying this module is based on the activity history, which reflects the amount of time (expressed for instance in minutes) which has elapsed since the start of the last episode of this activity type. It is assumed here that the larger the activity history - i.e. the longer ago the activity was executed for the last time -, the larger the reward obtained when selecting this activity. The functional form of this alternative reward function is inspired on the maximum attainable utility of the selected activity specified in Joh *et al.* (2004). The relationship between the activity history and the reward of this activity



can be expressed by following function:

$$R = R_{min} + \frac{R_{max}}{1 + e^{\beta(\alpha-h)}}. \quad (4.6)$$

Here  $h$  is the activity history,  $R$  is the reward,  $R_{min}$  is the minimum reward and  $R_{max}$  is the maximum attainable reward for the activity. This reward function is a symmetrical  $S$ -shaped curve and resembles the reward function described by Equation 4.1. As is the case for the utility function in the duration module, the double logistic model is also estimated:

$$R = \frac{R_{max}^{(1)}}{1 + e^{\beta^{(1)}(\alpha^{(1)}-h)}} + \frac{R_{max}^{(2)}}{1 + e^{\beta^{(2)}(\alpha^{(2)}-h)}}. \quad (4.7)$$

The parameters  $R_{max}^{(1)}$ ,  $\beta^{(1)}$  and  $\alpha^{(1)}$  determine the lower  $S$  of the curve, while the parameters  $R_{max}^{(2)}$ ,  $\beta^{(2)}$  and  $\alpha^{(2)}$  fix the upper  $S$  of the curve. Both functions are fitted if at least 20 cases are observed in the data. The parameters of the best fit for each activity type for cluster one are summarized in table 4.11.

Activity type	Functional form	Parameters	Plot
In-home activities	Logistic	$\alpha = 638.3798$ $\beta = 0.0097$ $R_{min} = 0.0228$ $R_{max} = 0.9034$	
Working	Double logistic	$\alpha^{(1)} = 1290.5521$ $\beta^{(1)} = 0.0059$ $R_{max}^{(1)} = 0.7814$ $\alpha^{(2)} = 4318.3857$ $\beta^{(2)} = 0.0165$ $R_{max}^{(2)} = 0.1688$	

Continued on Next Page...

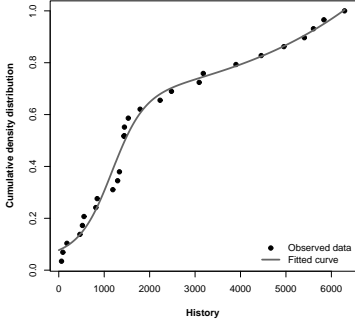
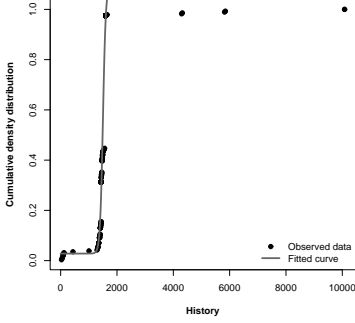
Activity type	Functional form	Parameters	Plot
Bring/Get	Double logistic	$\alpha^{(1)}$ = 1166.7160	
		$\beta^{(1)}$ = 0.0028	
		$R_{max}^{(1)}$ = 0.5930	
		$\alpha^{(2)}$ = 15,788.2653	
		$\beta^{(2)}$ = 0.0003	
		$R_{max}^{(2)}$ = 9.2774	
Sleeping	Logistic	$\alpha$ = 1502.1829	
		$\beta$ = 0.0197	
		$R_{min}$ = 0.0281	
		$R_{max}$ = 1.0695	

Table 4.11: Parameters of the best curve fitted to the observed data of cluster 1

These functions do not require being transformed into progress estimators as described in section 4.2.1, because the current module does not deal with a distant goal for which the reward is only assigned when this goal is reached. Furthermore, in case an activity type is observed less than 20 times in the training data, these reward curves can not be estimated accurately and the reward function is then based on the activity history  $h$  and the quantiles  $q_{25}$ ,  $q_{50}$  and  $q_{75}$  of the history of the observed activity, with probabilities 25%, 50% and 75% respectively. The actual reward function looks as follows:

$$R = \begin{cases} 0 & \text{if } h < q_{25}, \\ \frac{0.50-0.25}{q_{50}-q_{25}} * (h - q_{25}) + 0.25 & \text{if } q_{25} \leq h < q_{50}, \\ \frac{1.00-0.50}{q_{75}-q_{50}} * (h - q_{50}) + 0.50 & \text{if } q_{50} \leq h < q_{75}, \\ 1 & \text{if } h \geq q_{75}. \end{cases} \quad (4.8)$$

An example of this reward function for the grocery shopping activity can be found in figure 4.11.

Table 4.12 lists the parameters used for this alternative history reward for cluster 1.

Activity type	$q_{25}$	$q_{50}$	$q_{75}$
In-home activities	-	-	-
Working	-	-	-
Services	7405	9925	10080
Out-of-home eating	1440	4010	5040
Grocery shopping	2443	4725	7808
Non-daily shopping	4325	7605	8610
Education	1075	2910	4320
Social activity	1655	2930	5415
Leisure	1555	4770	7560
Bring/Get	-	-	-
Touring	NA	NA	NA
Other	NA	NA	NA
Sleeping	-	-	-

Table 4.12: Parameters for the alternative reward functions in module 3 based on the observed data of cluster 1

#### 4.4.2 Validation

This section validates the functioning of the second and third module. To this purpose, the duration module is trained first. Thereafter, the second and third modules are trained

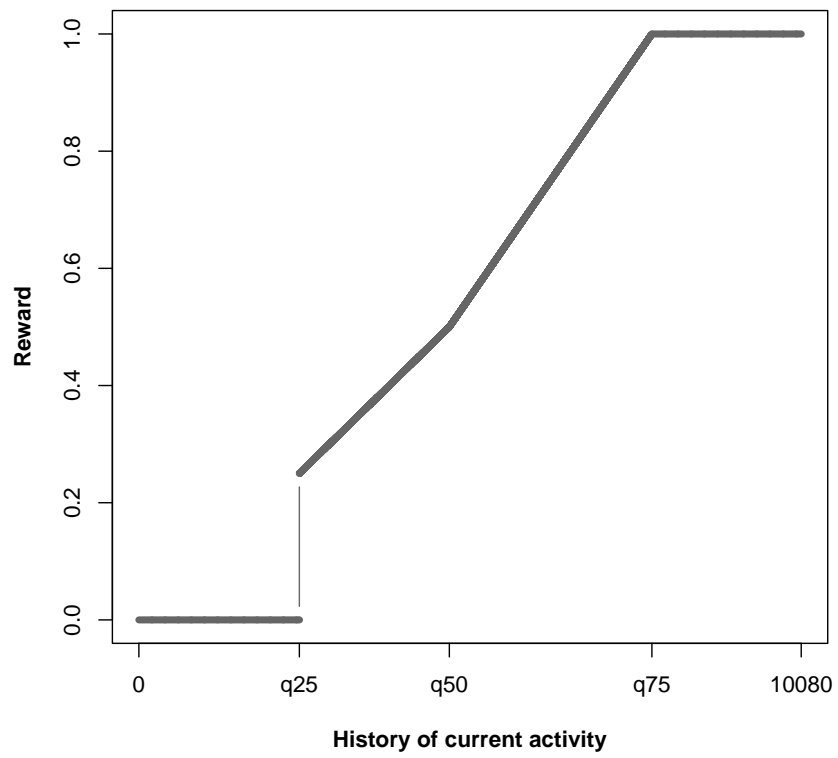


Figure 4.11: Alternative reward function in module 3 for the grocery shopping activity of cluster 1

simultaneously. These training phases both take 5,000 learning episodes. Additionally, to demonstrate the full potential of the suggested framework, these modules are interconnected by means of a multi-actor reinforcement learning framework as discussed in section 2.4.

The functioning of the second module, determining whether or not the next fixed activity is executed, is analysed first. To this end, a closer look on the action of this module is taken. Table 4.13 records for each cluster the number of times that the agent decides to perform the next fixed activity (F=Y) as opposed to proceeding with an alternative activity (F=N), either inside (Inside fixed time slot (FTS)) or outside (Outside fixed time slot (FTS)) the predetermined time slot for the starting time of this fixed activity (i.e.  $[EST, FST]$ ). This table indicates that, inside the fixed time slot, the agent always chooses to execute the next fixed activity (22% vs. 0%). Outside this fixed time slot, the agent does not want to turn to this next fixed activity in the majority of the cases (76% vs. 3%).

		<b>F=N</b>		<b>F=Y</b>	
<b>Cluster 1</b>	<b>Inside FTS</b>	0	0%	82	26%
	<b>Outside FTS</b>	222	71%	9	3%
<b>Cluster 2</b>	<b>Inside FTS</b>	0	0%	66	17%
	<b>Outside FTS</b>	312	80%	14	4%
<b>Cluster 3</b>	<b>Inside FTS</b>	0	0%	23	25%
	<b>Outside FTS</b>	70	75%	0	0%
<b>All</b>	<b>Inside FTS</b>	0	0%	172	22%
	<b>Outside FTS</b>	604	76%	23	3%

Table 4.13: Validation results for the fixed activity selection module 2

Subsequently, the results generated by modules 2 and 3 are examined in table 4.14 for the reward scheme based on the sequence alignment method, and in table 4.15 for the history-based reward functions. These tables compare the simulated sequences to the sequences observed in the validation set, first for each cluster separately and subsequently for the entire set. To start with, the average and standard deviation of the number of activities in the sequences is calculated. Next, the dissimilarity of the sequences in the simulated set and in the validation set is determined by means of a sequence alignment method, for which the penalty cost for insertion and deletion is set to one unit, while the penalty cost for substitution is set to two units. Finally, this sequence alignment measure is also used

to estimate the dissimilarity between each simulated sequence and its counterpart in the validation set.

	Cluster 1	Cluster 2	Cluster 3	All
Sequence length of validation sequences				
Avg.	6.04	4.41	3.57	<b>4.80</b>
St.dev.	1.35	2.07	1.04	<b>1.95</b>
Sequence length of simulated sequences				
Avg.	6.70	5.03	4.00	<b>5.40</b>
St.dev.	0.55	0.16	0.00	<b>1.02</b>
SAM in set of validation sequences				
Avg.	0.37	0.41	0.55	<b>0.51</b>
St.dev.	0.23	0.30	0.39	<b>0.28</b>
SAM in set of simulated sequences				
Avg.	0.35	0.01	0.20	<b>0.46</b>
St.dev.	0.14	0.04	0.31	<b>0.32</b>
SAM between validation sequences and simulated sequences				
Avg.	0.44	0.54	0.55	<b>0.51</b>
St.dev.	0.17	0.10	0.32	<b>0.18</b>
t-statistic	2.63	10.72	-0.05	<b>-0.35</b>
P-value	0.0055	0.0000	0.5182	<b>0.6355</b>

Table 4.14: Validation results for the activity selection modules 2 and 3 generated by means of the SAM-based reward function

Next to the average and standard deviation of these sequence alignment indicators, the dissimilarity between the simulated sequences and their corresponding validation sequences is put against the average dissimilarity in the validation set. For this purpose, a one-tailed Student's  $t$ -test is applied to prove whether the dissimilarity estimated between the simulated sequences and the validation sequences substantially exceeds the dissimilarity measured in the observed validation sequences. The hypotheses supporting this test equal:

$$\begin{aligned}
 H_0 &: \mu_{validation} = \mu_{validation \text{ vs simulated}}, \\
 H_1 &: \mu_{validation} < \mu_{validation \text{ vs simulated}}.
 \end{aligned}
 \tag{4.9}$$

In this case, rejecting the null hypothesis signifies that the simulated sequences differ significantly more from the validation sequences, compared to the differences between the

	Cluster 1	Cluster 2	Cluster 3	All
Sequence length of validation sequences				
Avg.	6.04	4.41	3.57	4.80
St.dev.	1.35	2.07	1.04	1.95
Sequence length of simulated sequences				
Avg.	7.91	8.72	4.30	7.78
St.dev.	1.23	1.49	0.63	2.02
SAM in set of validation sequences				
Avg.	0.37	0.41	0.55	0.51
St.dev.	0.23	0.30	0.39	0.28
SAM in set of simulated sequences				
Avg.	0.60	0.85	0.35	0.88
St.dev.	0.17	0.26	0.36	0.27
SAM between validation sequences and simulated sequences				
Avg.	0.62	0.87	0.54	<b>0.74</b>
St.dev.	0.21	0.18	0.27	<b>0.25</b>
t-statistic	8.20	21.97	-0.22	<b>11.10</b>
P-value	0.0000	0.0000	0.5861	<b>0.0000</b>

Table 4.15: Validation results for the activity selection modules 2 and 3 generated by means of the history-based reward functions

sequences within the validation set. Not rejecting the null hypothesis may indicate that the difference between the simulated sequences and their matching validation sequences is not larger than the dissimilarity within the validation set.

These tables show that the algorithm implementing the sequence alignment-based reward scheme outperforms the algorithm using the history-based reward functions, as reflected in the low sequence alignment values matching the variability in the validation set. However, a comment should be given with regard to these results: given that the reward function guiding the learning process of the former reinforcement learning agent is based on the sequence alignment measure which is also used to validate its outcome, this reward scheme obviously enables generating sequences which score better with respect to this criterion.

Therefore, additional statistics, displayed in tables 4.16 and 4.17, are calculated to compare the results of the sequence alignment-based and history-based reward functions. These statistics also support the conclusions drawn from the validation results based on the sequence alignment measure. As the outcome of the algorithm utilizing the history-based reward functions is not satisfactory, the reward scheme based on the sequence alignment



method is used throughout the remainder of this work.

	<b>SAM-based</b>	<b>History-based</b>	<b>Test</b>
In-home activities	100.00	94.59	96.62
Working	31.76	45.95	34.46
Services	4.05	17.57	2.03
Out-of-home eating	4.05	26.35	12.16
Grocery shopping	3.38	33.78	7.43
Non-daily shopping	0.00	41.89	4.73
Education	3.38	29.05	4.05
Social activity	4.05	37.16	16.89
Leisure	14.86	47.97	7.43
Bring/Get	54.05	83.11	10.14
Touring	4.73	19.59	0.00
Other	5.41	13.51	0.00
Sleeping	100.00	100.00	100.00

Table 4.16: Validation results for the activity selection modules 2 and 3: relative frequency (%) of sequences containing the specified activity type

	<b>SAM-based</b>	<b>History-based</b>	<b>Test</b>
In-home activities	2.09	1.41	1.71
Working	1.06	1.22	1.20
Services	1.00	1.04	1.00
Out-of-home eating	1.00	1.05	1.17
Grocery shopping	1.00	1.16	1.00
Non-daily shopping	NA	1.06	1.00
Education	1.00	1.14	1.33
Social activity	1.00	1.04	1.04
Leisure	1.00	1.13	1.00
Bring/Get	1.00	1.20	1.40
Touring	1.00	1.17	NA
Other	1.00	1.25	NA
Sleeping	1.99	1.93	2.01

Table 4.17: Validation results for the activity selection modules 2 and 3: average number of episodes of the specified activity type within the sequences containing this activity type

## 4.5 Module 4: Location

### 4.5.1 Reward Function

In the fourth module the agent picks the location on which the selected activity takes place. For this decision, the agent chooses from a set of seven locations. Location zero is the home

location, while locations one to six correspond to a distance band counted from the home location; the boundaries of these distance bands are: less than 2 km (but not the home location), 2 – 5 km, 5 – 10 km, 10 – 25 km, 25 – 50 km and more than 50 km respectively. The reward of the location decision is assigned based on the location preferences for the specified activity type as inferred from probability distribution of the underlying observed diary data. Tables 4.18 to 4.20 show the observed preferences as estimated from the observed data for cluster 1 to 3 respectively.

Activity type	Home	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6
In-home activities	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
Working	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000
Services	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
Out-of-home eating	0.0000	0.0000	0.1579	0.1579	0.2632	0.2632	0.1579
Grocery shopping	0.0000	0.5455	0.0909	0.1818	0.1818	0.0000	0.0000
Non-daily shopping	0.0000	0.1111	0.2222	0.1111	0.2222	0.2222	0.1111
Education	0.0000	0.0000	0.3846	0.0000	0.0769	0.3846	0.1538
Social activity	0.0000	0.1333	0.2000	0.1333	0.2000	0.0667	0.2667
Leisure	0.0000	0.2308	0.1538	0.3077	0.1538	0.0769	0.0769
Bring/Get	0.0000	0.2414	0.4138	0.1379	0.1034	0.1034	0.0000
Touring	-	-	-	-	-	-	-
Other	-	-	-	-	-	-	-
Sleeping	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000

Table 4.18: Parameters for the reward functions in module 4 based on the observed data of cluster 1

Activity type	Home	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6
In-home activities	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
Working	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000
Services	0.0000	0.2000	0.2000	0.2500	0.2000	0.1000	0.0500
Out-of-home eating	0.0000	0.4286	0.0714	0	0.2143	0.0714	0.2143
Grocery shopping	0.0000	0.4483	0.4138	0.0690	0.0000	0.0345	0.0345
Non-daily shopping	0.0000	0.2727	0.3182	0.2727	0.0455	0.0909	0.0000
Education	0.0000	0.1111	0.0000	0.3333	0.0000	0.2222	0.3333
Social activity	0.0000	0.1515	0.2121	0.2121	0.2424	0.1212	0.0606
Leisure	0.0000	0.2759	0.1034	0.2069	0.1379	0.0345	0.2414
Bring/Get	0.0000	0.3585	0.0755	0.4717	0.0566	0.0000	0.0377
Touring	-	-	-	-	-	-	-
Other	-	-	-	-	-	-	-
Sleeping	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000

Table 4.19: Parameters for the reward functions in module 4 based on the observed data of cluster 2

Activity type	Home	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6
In-home activities	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
Working	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000
Services	-	-	-	-	-	-	-
Out-of-home eating	0.0000	0.0000	0.5000	0.0000	0.5000	0.0000	0.0000
Grocery shopping	-	-	-	-	-	-	-
Non-daily shopping	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
Education	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
Social activity	0.0000	0.2000	0.0000	0.2000	0.0000	0.0000	0.6000
Leisure	0.0000	0.0000	0.2500	0.2500	0.2500	0.0000	0.2500
Bring/Get	0.0000	0.6667	0.0000	0.3333	0.0000	0.0000	0.0000
Touring	-	-	-	-	-	-	-
Other	-	-	-	-	-	-	-
Sleeping	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000

Table 4.20: Parameters for the reward functions in module 4 based on the observed data of cluster 3

The execution of in-home activities is restricted to the home location, which does not demand any further explanation. In addition, the current research assumes that the fixed activities can occur at only one location as well. For the sleeping activity, this location is set to the home location, whereas the location for the working activity is inferred from the observed data. The fixed working location is defined to be the location with the highest probability in the observed sequences. This concept of fixed locations is reflected in the reward function by assigning a rather high reward for executing the activity at the fixed location, while determining a large penalty for executing the activity at alternative location as shown in tables 4.18 to 4.20. Consequently, the location of a fixed activity does not have to be pre-determined and is selected by the agent based on the reward accumulated in the course of the learning algorithm. The fixity of the location is thus supported by the underlying reward function.

Furthermore, it is worth noting that the decision of this module is of major influence on the selection of the travel mode used to get to this location in the next module. This interaction between the decision modules brings two advantages of reinforcement learning and its multi-actor version to the surface.

First, a reinforcement learning agent does not only learn to select the most optimal action in a given state, he also learns the value of all feasible actions in this state. This characteristic allows the agent to choose the best action - i.e. the action corresponding to the largest value -, but it also enables indicating the second best action in case the best action is not attainable for some reason. For example, presume that the most preferred location for performing a certain activity is location six (i.e. a location more than 50 km from the home location). However due to car unavailability, this location may not be reached and is thus infeasible. In this case, the reinforcement learning agent does not have to be retrained and is able to select the second optimal location for this activity type autonomously and without any additional effort.

This leads to the second advantage; one linked to the multi-actor variant of reinforcement learning. Because the decision modules are strongly interrelated, a high-level decision (in this case selection of the activity location) is very likely to influence the value of a lower-level decision (in this case selection of the travel mode) to a large extent. Yet, if the high-

level decision module does not take the value of its subordinate decisions modules into consideration, the reinforcement learning system may get stuck in a suboptimal solution. For this purpose, multi-actor reinforcement learning is applied here as described in section 2.4. A multi-actor reinforcement learning agent ensures that the order in which the decisions are taken does not affect the value of the optimal solution selected. After all, a multi-actor reinforcement agent can select an apparently less optimal action in one module in order to be able to reach an action in a next module, which results in a higher global action value.

#### **4.5.2 Validation**

To verify the functioning of the current module, the duration module is trained first, after which the second and third modules determining the sequence of activities are trained, and finally the location module is trained. The number of learning episodes for each training phase equals 5,000. Moreover, as is the case for the previous modules, this module is also incorporated into the learning framework by means of multi-actor reinforcement learning to enable interaction between the decision levels.

Table 4.21 summarizes for each activity type the location that is selected most. The results presented here are very favourable: when putting this table against the observed location preferences displayed in tables 4.18 to 4.20, it can be concluded that the reinforcement learning algorithm is able to reproduce these preferences very well. Furthermore, all in-home and all sleeping activity episodes in the simulated sequences take place at the home location, and all working activity episodes occur at the pre-determined fixed work location.

### **4.6 Module 5: Travel mode**

#### **4.6.1 Reward Function**

The fifth and final module is held responsible for selecting the transport mode used to reach the desired location in case this location differs from the location of the previous activity. In the current research, the travel mode choice set consists of a slow mode (i.e. on foot, by bike or by moped), a private motorized vehicle (i.e. by motorcycle or by car, either as driver or as passenger) or public transport (i.e. by train, by bus, by taxi or by subway), called “slow”,

<b>Activity type</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>
In-home activities	0	0	0
Working	5	-	-
Services	4	-	-
Out-of-home eating	5	-	-
Grocery shopping	1	1	
Non-daily shopping	4	-	-
Education	5	-	-
Social activity	4	-	-
Leisure	3		2
Bring/Get	3	-	-
Touring	1/2	-	-
Other	2	-	-
Sleeping	0	0	0

Table 4.21: Validation results for the location module 4

“car” and “public” respectively.

In this module, the agent receives a reward which depends on the travel time required to get to the selected location and the travel mode of the current home-based tour (if any). If the selected travel mode does not equal the travel mode of the current home-based tour, the agent receives a penalty if he switches from the slow mode or public transportation to a private motorized vehicle or vice versa. If the travel mode of the home-based tour is set to the slow mode and the agent chooses to use the public transportation to get to the desired location or the other way round, the agent does not receive this penalty due to the interchangeability of these transport modes.

The reward based on the travel time is inspired by the concept of a constant travel time budget (Zahavi & Talvitie, 1980; Zahavi & Ryan, 1980), which suggests that people are willing to travel a certain amount of time to reach a certain location to execute a particular activity. This idea implies that, if - for one reason or another - the speed of the habitual travel mode used increases or if a particular individual uses an alternative travel mode, both of which enable spanning a larger distance in the same amount of time, this individual is prepared to travel further to execute the same activity; causing the individual’s sphere of action to expand accordingly (Hägerstrand, 1970). In this perspective, the reward of the fifth module increases linearly towards the highest attainable reward for this module when the travel time to reach the selected location by means of the chosen travel mode approaches



the average of the observed travel times for the current activity. The reward decreases when the required travel time drifts away from this average. Figure 4.12 displays the course of this reward function.

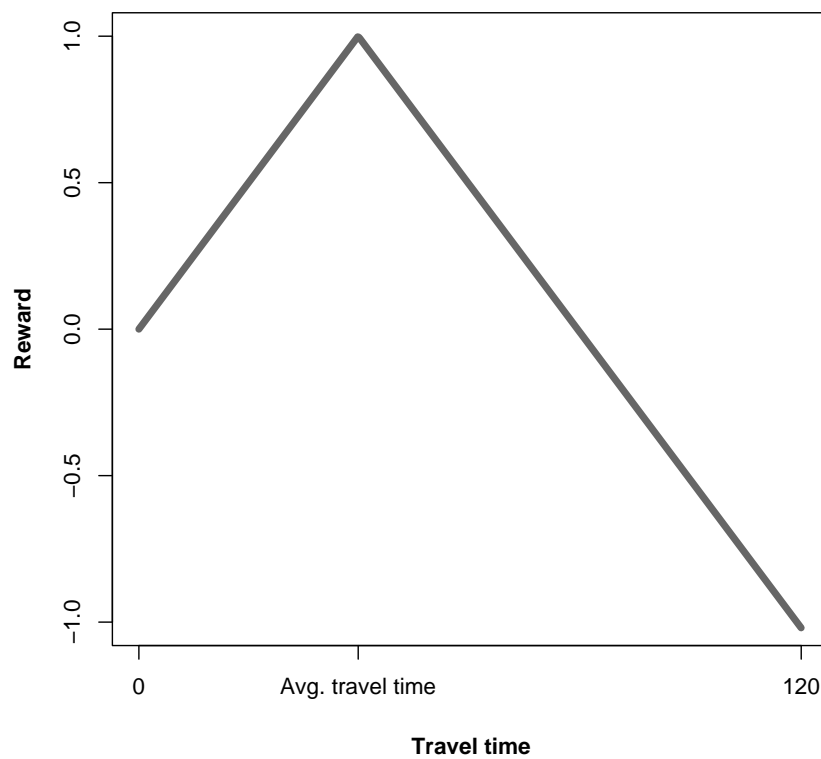


Figure 4.12: Reward function in module 5 for the working activity of cluster 1

#### 4.6.2 Validation

As the choice of the transport mode is highly interrelated with the preceding activity and location choices, as well as with the travel mode of the current out-of-home tour, the validation of the fifth module is included in the validation of the entire scheduling framework. For this purpose, the generated activity-travel patterns are compared to their corresponding counterparts in the validation set by means of a multidimensional sequence alignment method

(DP-SAM) - as proposed in Joh *et al.* (2001) - based on the activity type, location category and travel mode. In this case, the technique is suited as all dimensions which are compared, contain categorical data. Furthermore, it is assumed here that the distance between two categories is independent of these categories, which is particularly important for comparing the location dimension.

To calculate the multidimensional dissimilarities between the sequences, the dimensions are weighted as follows: the activity type receives a weight of two, while the location category and travel mode get a weight of one. Tables 4.22 and 4.23 give an overview of the outcome of this comparison in the short and long format respectively. The hypotheses for the  $T$ -tests in these tables resemble the ones specified in Equation 4.9.

	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>All</b>
SAM in set of validation sequences				
Avg.	1.80	1.72	2.16	<b>1.91</b>
St.dev.	0.57	0.76	0.77	<b>0.68</b>
SAM in set of simulated sequences				
Avg.	1.07	0.23	1.80	<b>1.19</b>
St.dev.	0.38	0.25	0.65	<b>0.70</b>
SAM between validation sequences and simulated sequences				
Avg.	2.08	1.92	2.68	<b>2.09</b>
St.dev.	0.48	0.54	0.79	<b>0.62</b>
t-statistic	3.86	3.30	3.07	<b>3.51</b>
P-value	0.0002	0.0007	0.0025	<b>0.0003</b>

Table 4.22: Validation results for DP-SAM calculated based on the short format sequences simulated by the entire multi-actor reinforcement learning system

With regard to the comparison of the sequences in their short format, it catches the eye that for each cluster the distances measured in the set of validation sequences exceeds this distance measured in the set of simulated sequences. This observation indicates that the simulated sequences show less variability than the actual observed sequences, which is also endorsed by the lower standard deviations. This is particularly the case for the sequences simulated in cluster 2, for which the average DP-SAM is as low as 0.23. It should be noted here that, depending on the goal of the simulation, a low variability within the simulated sequences can be considered a drawback. For instance, if one is mainly interested in analysing the results at the level of the individual agent, rather than concentrating on

	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>All</b>
SAM in set of validation sequences				
Avg.	0.34	0.25	0.40	<b>0.38</b>
St.dev.	0.08	0.13	0.13	<b>0.13</b>
SAM in set of simulated sequences				
Avg.	0.16	0.01	0.26	<b>0.26</b>
St.dev.	0.07	0.01	0.11	<b>0.20</b>
SAM between validation sequences and simulated sequences				
Avg.	0.39	0.26	0.48	<b>0.33</b>
St.dev.	0.09	0.08	0.07	<b>0.12</b>
t-statistic	3.66	0.26	4.16	<b>-5.02</b>
P-value	0.0003	0.3975	0.0001	<b>1.000</b>

Table 4.23: Validation results for DP-SAM calculated based on the long format sequences simulated by the entire multi-actor reinforcement learning system

aggregated results, a low variability usually indicates that all sequences belonging to the same cluster are (virtually) the same, which may distort the results of subsequent analyses. Yet, the variability of the simulated sequences within the same cluster can be influenced by a number of operations, such as increasing the number of prototypes so that less agents from the synthetic population correspond to the same prototype agent or redesigning the reward functions to allow for more variability.

When comparing the simulated sequences with their corresponding observed validation sequences, the average DP-SAM is higher than the average DP-SAM in the validation set. However, at the 0.01%-level none of these differences are significant. Considering these results for the multidimensional sequences in the long format presented in table 4.23, the average of the dissimilarities between the validation and simulated sequences for the entire validation set is actually smaller than the average dissimilarity of the sequences in the validation set. These results indicate that the simulated sequences match the corresponding validation sequences well and that the presented framework is thus highly suited to reproduce individual activity-travel patterns based information distilled from observed activity-travel sequences. These results are elaborated and fine-tuned further in the next chapter.

## 4.7 Conclusions

This chapter describes the scheduling engine, founding the core of the research presented in this manuscript, based on multi-actor reinforcement learning with regression tree function approximation. In general, the scheduling process is conceived as follows: at a certain time of the day, all agents are in a state and are performing an activity (or travel related to this activity), which is characterized amongst other attributes by the duration of this activity up till the current time slot, its location and travel mode. Before proceeding to the next time step, each agent decides either to continue executing the current activity for one more time step, or to start executing another activity, in which case the agent selects the activity type, activity location and travel mode used to get to this location. To realize this concept, the scheduling engine contains five modules, each of which corresponds to one activity-travel related decision.

The first module is responsible for determining the duration of the activity that is being executed by deciding either to continue or stop executing the current activity. The reward guiding the action selection of the corresponding reinforcement learning module is based on the duration of each activity type observed in the data set. If the agent stops performing the current activity, the agent proceeds to the next modules.

To account for the existence of fixed activities, such as working and sleeping, the agent chooses whether or not to start the next fixed activity in the second module. The reward accompanying this decision depends on the time left to the earliest and last possible starting times of the fixed activity.

In case the agent does not want to commence this fixed activity, he decides on which activity he does want to carry out in the third module. In the present chapter, two reward functions supporting this decision are tested: the first reward function is based on the history of the selected activity, while the second one reflects the improvement in the unidimensional sequence alignment measure when adding the selected activity to the existing sequence with respect to the observed sequences. The analyses in this chapter demonstrated that the sequence alignment-based reward is preferred to the history-based reward function.

Next, the agent selects a distance band from his home location, defining the location where he wants to perform the selected activity, either fixed or flexible, in the fourth module.

The reward of this decision is founded on the attributes describing the locations in the observed data.

Finally, the fifth reinforcement learning module accounts for selecting the travel mode to get to the desired location, in case this location does not equal the home location. In this module, the reward is estimated based on the observed travel time budgets.

For each of these modules, the present chapter elaborates on the design of the reward functions and illustrates the functioning and validity of these modules. Finally, the results presented in this chapter indicate that the described scheduling framework reaches its postulated goals and thus is capable of simulating activity-travel sequences based on observed diary data. The performance of this framework is analysed in the next chapter.



## Chapter 5

# Performance

### 5.1 Introduction

The ultimate goal of the scheduling framework described in this dissertation is to be fitted in an activity-based modelling system. Consequently, the ability of this scheduling algorithm to generate daily activity-travel sequences within a realistic experimental setting has to be evaluated. After all, the results presented in the previous chapter - though very promising - are generated based on a small-scale experiment and do not take into account the usability of the current scheduling framework in modelling the activity-travel behaviour of a realistic population group, which for instance contains more than 5 million individuals, as is the case for the current study area Flanders (Belgium) (in 2008 Flanders consisted of 6,161,600 inhabitants; 4,944,809 of which are adult inhabitants<sup>1</sup> and form the target population of the present research).

Therefore, this chapter runs through the entire framework discussed in chapter 4 and visualized in figure 4.1. The current chapter examines the computational performance of the algorithm in section 5.2 and suggests a number of actions to improve and accelerate the scheduling algorithm in section 5.3. To end with, these improvements are assessed in section 5.4.

---

<sup>1</sup>Source: <http://www4.vlaanderen.be/>

## 5.2 Analysis of the Performance of the Algorithm

### 5.2.1 Preliminary Operations

The previous chapter shows that the scheduling algorithm is capable of predicting activity-travel diaries on a very detailed level - i.e. 5-minute basis, 13 activity categories, 7 feasible locations and 3 travel modes. However, based on the available data, it should be considered to drop the level of detail of one or more dimensions to improve the prediction quality. After all, taking a closer look at the statistics of the observed data as presented in tables 4.3 to 4.5, one will notice that some activity categories do not provide enough instances to support a substantiated training of the scheduling algorithm. For instance, with respect to the service activity only 4 episodes in cluster 1, 20 episodes in cluster 2 and not even one episode in cluster 3 are observed. Consequently, the 13 activity types in the original data set are aggregated and reclassified into five categories: in-home activities, sleeping, working, mandatory activities and discretionary activities. The in-home and sleeping activity categories are adopted from the original activity categories as these activity types occur very frequently in the observed data set and do not raise any difficulties. The working activity is also copied from the original activity categories as this activity is considered to be fixed in time and space, and cannot be put together with more flexible activity types. Furthermore, the working activity is observed frequently enough in the data set to act as stand-alone category. The category of mandatory activities joins the service activities, daily shopping, education and bring/get somebody/something, as they are all supposed to be less susceptible to delays and substitution by other activities and tend to be more fixed compared to the activity categories out-of-home eating, non-daily shopping, social activities, leisure touring and other activities, which are now labelled discretionary activities. The effect on the number of observed episodes for each activity type is reflected in table 5.1. The careful reader will notice that the number of observed mandatory and discretionary activity episodes do not equal the sum of the observed instances of their compounding former activity categories. This is due to the fact that subsequent activity episodes covering the same activity type are merged. Taking into account the fact that reducing the level of detail impacts the speed of the algorithm, this step is contemplated prior to analysing the performance of the algorithm.



The performance analyses thus proceed based on this new activity classification, while the levels of the dimensions activity location and travel mode remain unchanged.

Old activity type	#	New activity type	#
In-home activities	996	In-home activities	996
Sleeping	1193	Sleeping	1193
Working	227	Working	227
Services	27	Mandatory activities	204
Grocery shopping	51		
Education	31		
Bring/Get	106		
Out-of-home eating	56	Discretionary activities	209
Non-daily shopping	39		
Social activity	79		
Leisure	57		
Touring	0		
Other	0		

Table 5.1: Number of activity episodes observed for the old and new activity classification

The modified data serve as input to the scheduling algorithm. Aiming at walking through and evaluating the entire agent-based micro simulation framework summarized in figure 4.1, the dataset is split into a training and validation set of 75% and 25% respectively at this point in time. For illustration purposes, the observed population corresponds to the training set, whereas the synthetic population corresponds to the validation set. This approach does not entirely match an actual simulation process, in which the observed population generally contains all observed activity-travel diaries, and the synthetic population is constructed as such that each individual within the synthetic population matches a particular individual of the target population (i.e. the population for which a simulation of activity-travel sequences is desired).

In the first step of the prediction process, the activity-travel diaries contained in the training set are converted to the long sequence format in order to calculate dissimilarities between these training sequences. Next, the clustering algorithm described in 3.4.1 is applied to these dissimilarities. As a result, three clusters of sequences displaying similar activity-travel behaviour can be identified based on the silhouette width.

Some general statistics on the sequence characteristics and the activities recorded in these

sequences for each of the clusters are included in tables 5.2 and 5.3, while tables 5.4 and 5.5 give an overview on the activity durations and table 5.6 examines the radius of action of the observed sequences. Tables 5.2 and 5.3 indicate that cluster 2 mainly consists of sequences including at least one working episode (95%), while only 5 and 4 percent of the sequences of clusters 1 and 3 respectively contain a working activity. Furthermore, both table 5.4 and table 5.5 show that the average duration spent on the working activity is considerably less for clusters 1 and 3, compared to cluster 2. Additionally, the majority of the sequences within cluster 3 (67%) only include an in-home activity in between two sleeping activities, explaining the large average duration per in-home activity episode (443 minutes) with respect to cluster 2. A rather large number of sequences in cluster 1 and 2 records at least one mandatory activity (25% and 30% respectively), whereas only a small number of sequences of cluster 3 includes this activity type (7%). Table 5.5 reveals that in cluster 1 more time is spent in the course of the entire observed sequences on in-home activities (860 minutes) with respect to the other two clusters and in cluster 3 more time is spent in total on discretionary activities (344 minutes). Finally, table 5.6 takes a look at the location of the activity which took place in the highest distance band with respect to the home location. This table shows that the sequences of clusters 1 and 3 are less geographically dispersed than the sequences of cluster 2.

	Cluster 1	Cluster 2	Cluster 3
# of sequences	232	147	67
Avg.# of activities in sequences	4.58	5.90	3.28
% of sequences which match the following patterns:			
Sleep-Home-Sleep	50.86	0.68	67.16
Sleep-Home-{Some activities,including work}-Home-Sleep	3.45	71.43	0.00
Sleep-Home-{Some activities,not including work}-Home-Sleep	33.62	3.40	5.97

Table 5.2: Some general descriptive statistics of the clusters

	Cluster 1	Cluster 2	Cluster 3
<b>In-home activities</b>	100.00	100.00	88.06
<b>Sleeping</b>	100.00	100.00	100.00
<b>Working</b>	4.74	95.24	4.48
<b>Mandatory activities</b>	25.43	30.61	7.46
<b>Discretionary activities</b>	31.03	33.33	22.39

Table 5.3: Percentage of sequences in clusters containing the listed activities at least once

	Cluster 1			Cluster 2			Cluster 3		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>In-home activities</b>	394	507	383	302	209	191	63	443	322
<b>Sleeping</b>	467	459	95	294	442	57	134	450	47
<b>Working</b>	11	230	131	163	459	158	3	338	188
<b>Mandatory activities</b>	97	78	109	56	85	119	5	132	209
<b>Discretionary activities</b>	94	140	124	52	153	161	15	344	210

Table 5.4: Number of observed activity episodes and average and standard deviation of the duration of these activity episodes

	Cluster 1			Cluster 2			Cluster 3		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>In-home activities</b>	232	860	239	147	429	182	59	473	322
<b>Sleeping</b>	232	924	126	147	884	83	67	899	68
<b>Working</b>	11	230	131	140	535	107	3	338	188
<b>Mandatory activities</b>	59	129	139	45	106	144	5	132	209
<b>Discretionary activities</b>	72	183	136	49	162	174	15	344	210

Table 5.5: Number of sequences containing activities and average and standard deviation of the total duration of the activities within these sequences

	Cluster 1	Cluster 2	Cluster 3
<b>Home</b>	51.72	0.68	67.16
<b>&lt;2km</b>	8.62	2.04	7.46
<b>2-5km</b>	8.19	12.93	2.99
<b>5-10km</b>	12.93	12.93	4.48
<b>10-25km</b>	6.90	25.17	10.45
<b>25-50km</b>	6.03	24.49	2.99
<b>&gt;50km</b>	5.60	21.77	4.48

Table 5.6: Percentage of sequences in which the listed distance bands corresponds to the farthest location reached

In the next step, the regression tree determining the socio-demographic profiles for these clusters is fitted. The result is displayed in table 5.7. This regression tree shows that the main explanatory variables are the day of the week (weekdays versus weekend days), age and two work-related variables, in particular the registered number of working hours per week and the work schedule. As expected from the descriptive statistics above, cluster 2 covers sequences recorded on weekdays, the majority of which can be matched to full-time working individuals (i.e. working more than 20 hours per week) under the age of 55. The sequences of cluster 1 are mainly recorded on weekdays by non-working individuals or are recorded on weekend-days. The profile corresponding to the sequences of cluster 3, is not as easily distinguishable.

This raises one major disadvantage of the described scheduling framework: each prototype agent corresponds to one and only one cluster. For each of the test agents, the cluster number is obtained based on the regression tree as follows: the values of the independent variables - which are assumed to be known for the agents of the synthetic population and thus also for the agents of the test set - are used to determine the node of the regression tree to which the agent belongs. The agent is then assigned the cluster number that corresponds to the highest relative frequency in the resulting node. These relative frequencies are calculated by dividing the number of observed cases of the cluster into consideration and belonging to the examined node by the sum of the number of observed cases for all clusters assigned to this node.

However, some clusters are not prevailing enough to be selected in any of the leaves, which is also the case for cluster 3, as shown in the regression tree results presented in table 5.7. Furthermore, valuable information is lost for the scheduling algorithm when assuming that an agent is part of one and only one cluster. After all, as the relative frequencies differ for each node, agents belonging to different nodes should be processed differently in the scheduling algorithm according to these relative frequencies. This issue is taken on in the next section.

Test rule	NodeNr.	#cases	Pred.	%clu1	%clu2	%clu3	Leaf
<i>Root</i>	0	446	1	0.5202	0.3296	0.1502	
<i>Weekday : weekday</i>	1	307	2	0.2803	0.3206	0.0874	
<i>Work : no</i>	3	9	1	0.1682	0.0157	0.0381	
<i>Age &lt; 37</i>	5	22	1	0.0224	0.0112	0.0157	*
<i>Age &gt; 37</i>	6	74	1	0.1390	0.0045	0.0224	*
<i>Work : parttime, fulltime</i>	4	208	2	0.1121	0.3049	0.0493	
<i>Age &lt; 55.5</i>	7	201	2	0.1009	0.3049	0.0448	
<i>Workinghours &lt; 20.5</i>	9	15	1	0.0202	0.0090	0.0045	*
<i>Workinghours &gt; 20.5</i>	10	174	2	0.0695	0.2892	0.0314	*
<i>Age &gt; 55.5</i>	8	7	1	0.0112	0.0000	0.0045	*
<i>Weekday : weekend</i>	2	139	1	0.2399	0.0090	0.0628	
<i>Age &lt; 41.5</i>	11	46	1	0.0605	0.0000	0.0426	*
<i>Age &gt; 41.5</i>	12	92	1	0.1794	0.0067	0.0202	*

Table 5.7: Outcome of decision tree attaching socio-demographical profiles to the cluster results

### 5.2.2 A First Glance at the Performance of the Algorithm

For the purpose of mapping the performance of the algorithm, one individual from the test set is selected to execute the subsequent analyses. The individual is 39 years old, is married and works full-time in a 38-hour work schedule. The simulations aim at predicting this individual's activity-travel pattern on a weekday. According to the decision tree in table 5.7, the relative frequencies for this agent equal 0.1782, 0.7414 and 0.0805 for cluster 1, cluster 2 and cluster 3 respectively (i.e. absolute frequencies of 0.0695, 0.2892 and 0.0314 respectively). The time resolution equals 5 minutes and the prototype agent is trained in the course of 15,000 episodes and refined in the course of 30 episodes.

In addition, the reinforcement learning framework as discussed in the previous chapter is compared to a multi-actor reinforcement learning framework incorporating  $Q$ -tables. Because the former approach allows the reinforcement learning agent to generalize over the state and action variables, the resolution used for training the prototype agents is equal to 15 minutes. A run of the latter approach takes 455 seconds, while a run of the enhanced reinforcement learning approach lasts for 731 seconds as measured on a server equipped with a 2.99 GHz Intel(R)Xeon(R) processor and 16.0 GB of RAM. Tables 5.8 and 5.9 provide a more profound insight into the division of this execution time over the called functions and the functions called within these functions (i.e. child functions) for both simulations. For the sake of clarity, only the functions which are no constructors or destructors of classes within the program and the functions of which the total execution time including the execution time of the child functions takes up more than five percent, are included in these tables.

Table 5.8 shows that the algorithm incorporating  $Q$ -tables spends the majority of its time (91%) calculating the dissimilarity based on the sequence alignment method for determining the reward of module 3. With respect to the multi-actor reinforcement learning system incorporating a regression tree function approximator, the share of this function is considerably less (32%). However, the implementation of this function is equal for both reinforcement learning systems, implying that the enhanced reinforcement learning system spends relatively more time maintaining its regression tree function approximator, which is revealed in table 5.9. In this case, 60% of the execution time is used to update the regression tree. 12% of the total execution time - i.e. 20% of the execution time required for updating

Function Name	%Time Usage Function+Children
Main	100.00
· Simulation::run	· 98.67
· · Agent::updateQ	· · 96.80
· · · Agent::calculateRewardmodule3	· · · 91.30
· · · · Inputsequences::calculateSam	· · · · 91.25
· · · · · Sequence::calculateUnisam	· · · · · 91.12
· · · · · <i>Rest</i>	· · · · · 0.13
· · · · <i>Rest</i>	· · · · 0.05
· · · <i>Rest</i>	· · · 5.50
· · <i>Rest</i>	· · 1.87
· <i>Rest</i>	· 1.33

Table 5.8: Time usage of functions for the multi-actor reinforcement learning scheduler incorporating  $Q$ -tables

Function Name	%Time Usage Function+Children
Main	100.00
· Simulation::run	· 99.23
· · Agent::updateQ	· · 94.94
· · · { Module1::updateQ	· · · { 45.32
· · · { Module2::updateQ	· · · { 10.25
· · · { Module3-5::updateQ	· · · { 5.70
· · · · Tree::updateTree	· · · · 59.71
· · · · · Node::updateNode	· · · · · 34.98
· · · · · · Node::updateStats	· · · · · · 32.10
· · · · · · · Studentst::calculateAlpha	· · · · · · · 11.73
· · · · · · · Node::isLeaf	· · · · · · · 7.70
· · · · · · · <i>Rest</i>	· · · · · · · 12.67
· · · · · · <i>Rest</i>	· · · · · · 2.88
· · · · · Node::fitNode	· · · · · 6.53
· · · · · Tree::calculateExplorationrate	· · · · · 14.49
· · · · · · Node::getSumvarianceleaves	· · · · · · 8.15
· · · · · · Node::getNuminstancesleaves	· · · · · · 6.32
· · · · · · <i>Rest</i>	· · · · · · 0.02
· · · · · <i>Rest</i>	· · · · · 3.71
· · · · <i>Rest</i>	· · · · 1.56
· · · Agent::calculateRewardmodule3	· · · 32.18
· · · · Inputsequences::calculateSam	· · · · 32.14
· · · · · Sequence::calculateUnisam	· · · · · 32.10
· · · · · <i>Rest</i>	· · · · · 0.04
· · · · <i>Rest</i>	· · · · 0.04
· · · <i>Rest</i>	· · · 1.49
· · <i>Rest</i>	· · 4.29
· <i>Rest</i>	· 0.77

Table 5.9: Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator



the regression tree - is required to calculate the value of  $\alpha$  in the regression tree. Furthermore, 14% of the execution time is consumed in updating the value for the exploration rate applied in the reinforcement learning mechanism.

### 5.2.3 Streamlining of the Program Code

To enable the use of the proposed scheduling framework in a large-scale experiment, major inefficiencies of the program code are eliminated at this point in time. For instance, within the scope of this research, the function applying the unidimensional sequence alignment method as part of determining the reward of module 3 is optimized. Additionally, the linear search algorithm underlying the computation of the value for the updated  $\alpha$ -parameter is changed to a binary search algorithm to enhance the speed of this computation. Finally, the values founding the calculation of the exploration rate are maintained when updating the regression tree, rather than calculating them from scratch whenever required.

As a result of these actions, the execution time for the reinforcement learning system incorporating  $Q$ -tables drops to 100 seconds and for the enhance reinforcement learning system to 221 seconds. The breakdowns of these execution times are recorded in tables 5.10 and 5.11. The share of auxiliary functions - such as the functions used to read the parameters from files - grows due to the fact that the execution time of the main functions of the system has decreased. Nevertheless, for convenience of comparison, these auxiliary functions are not included in the tables.

The tables show a considerable drop in the share of some of the functions discussed in the previous paragraph: for instance the function utilizing the unidimensional sequence alignment method decreases to 43% in the reinforcement learning system incorporating  $Q$ -tables and to 5% in the enhanced reinforcement learning system. Moreover, some functions even disappear from these lists, as is the case for the function computing the value of the  $\alpha$ -parameter in the regression tree and the exploration rate for the reinforcement learning system because the execution time required for these functions dropped considerably due to the implementation of the code optimizations suggested in the previous paragraph.

Function Name	%Time Usage Function+Children
Main	100.00
· Simulation::run	· 72.65
· · Agent::updateQ	· · 65.75
· · · Module1::updateQ	· · · 9.32
· · · Agent::calculateRewardmodule3	· · · 43.75
· · · · Inputsequences::calculateSam	· · · · 43.02
· · · · · Sequence::calculateUnisam	· · · · · 42.74
· · · · · <i>Rest</i>	· · · · · 0.28
· · · · <i>Rest</i>	· · · · 0.73
· · · <i>Rest</i>	· · · 12.68
· · Agent::selectAction	· · 6.63
· · <i>Rest</i>	· · 0.27
· <i>Rest</i>	· 27.35

Table 5.10: Time usage of functions for the multi-actor reinforcement learning scheduler incorporating  $Q$ -tables after improving code efficiency

### 5.3 Suggestions to Scaling Up the Algorithm

The previous section lifted the veil concerning the potential impact of improving the code efficiency on the speed of the system. Although these improvements are quite favourable, some advances on the design of the system are also inevitable. Therefore, the remainder of this section discusses a number of suggestions which aim at scaling up the algorithm for use in a large-scale experimental setting.

#### 5.3.1 Increase in the Number of Prototype Agents

Firstly, to benefit from the information contained in the socio-demographic profiles, the notion of a single prototype agent for each separate cluster is abandoned. As such, the algorithm can take advantage of the assignment frequencies for each cluster as recorded in the decision tree. This idea behind this improvement is borrowed from the concept of fuzzy clustering, which is extensively described in Hoppner *et al.* (1999). Fuzzy set theory does not assume a hard partitioning of the data set, in which each case of the data set can be assigned to one and only one cluster. Instead, it introduces the principle of “degree of membership”, which indicates the extent to which an instance matches the characteristics of each cluster.

In this view, the relative frequencies included in the decision tree are considered to

---

5.3. SUGGESTIONS TO SCALING UP THE ALGORITHM

---

Function Name	%Time Usage Function+Children
Main	100.00
· Simulation::run	· 88.79
· · Agent::updateQ	· · 76.61
· · · { Module1::updateQ	· · · { 49.07
· · · { Module2::updateQ	· · · { 13.90
· · · { Module3-5::updateQ	· · · { 3.83
· · · · Tree::updateTree	· · · · 62.41
· · · · · Node::updateNode	· · · · · 33.66
· · · · · Node::updateStats	· · · · · 24.49
· · · · · Rest	· · · · · 9.17
· · · · · Node::fitNode	· · · · · 17.69
· · · · · Node::calculateStats	· · · · · 6.85
· · · · · Rest	· · · · · 10.84
· · · · · Rest	· · · · · 11.06
· · · · Rest	· · · · 4.39
· · · Agent::calculateRewardmodule3	· · · 5.56
· · · · Inputsequences::calculateSam	· · · · 5.35
· · · · · Sequence::calculateUnisam	· · · · · 5.27
· · · · · Rest	· · · · · 0.08
· · · · Rest	· · · · 0.21
· · · Rest	· · · 4.25
· · Agent::selectAction	· · 12.11
· · · { Module1::selectAction	· · · { 9.82
· · · { Module2-5::selectAction	· · · { 1.98
· · · · { Module1::selectBestaction	· · · · { 9.73
· · · · { Module2-5::selectBestaction	· · · · { 1.96
· · · · · Tree::getBestvalue	· · · · · 10.75
· · · · · Rest	· · · · · 0.94
· · · · Rest	· · · · 0.11
· · · Rest	· · · 0.31
· · Rest	· · 0.07
· Rest	· 11.21

---

Table 5.11: Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator after improving code efficiency

indicate membership degrees and are incorporated in the reward functions as follows:

$$R_i(s, a) = \frac{\sum_j [m_{ij} * R_j(s, a)]}{\sum_j m_{ij}}. \quad (5.1)$$

Here  $R_i$  is the reward of executing action  $a$  in state  $s$  for prototype agent  $i$ ,  $R_j(s, a)$  is the reward according to the parameters estimated for cluster  $j$  for performing this action as defined in the previous chapter, and  $m_{ij}$  is the membership degree of cluster  $j$  for prototype  $i$ , as determined by the corresponding node of the decision tree describing the socio-demographic profiles. Consequently, one prototype agent can be constructed for each feasible combination of membership degrees inferred from the decision tree.

### 5.3.2 Softmax Action Selection

Secondly, it is decided to replace the  $\epsilon$ -greedy action selection method. As elaborated on in section 2.2.1, the  $\epsilon$ -greedy action selection strategy is a valuable method of guiding the exploration-exploitation trade-off. But, when exploring, the algorithm selects an action randomly from all available actions, disregarding the previously experienced action values. This signifies that the odds of selecting the worst action - based on the action values - are the same as the probability of selecting the next best action. Obviously, for the learning system it is more favourable to explore the opportunities posed by higher valued actions. Moreover, when exploiting its gathered knowledge, the agent greedily selects the highest valued action. However, in some cases a number of actions could be regarded as a set of optimal actions based on their action values. As a result, selecting an action either uniformly randomly and independently of the action values or greedily, does not capitalize the information collected on all actions.

From that perspective, it is advantageous to incorporate a softmax action selection strategy in which a selection probability is attached to all available actions according to their action values. The current research opts to use a Boltzmann distribution to rank and weight the actions (Sutton & Barto, 1998). The probability of selecting an action  $a_i$  equals:

$$\frac{e^{Q(s, a_i)/\tau}}{\sum_j e^{Q(s, a_j)/\tau}}. \quad (5.2)$$

Here  $\tau$  is a positive parameter and is called the temperature. When  $\tau$  is close to zero, the softmax action selection approaches the greedy action selection, whereas high temperatures cause the probabilities of the actions to be equal (Sutton & Barto, 1998). To steer the reinforcement learning agent, the  $\tau$ -value is generally set to be high at the start of the learning process and to decrease thereafter. As a result, the reinforcement learning agent tends towards a true exploration of the set of feasible actions at first and is inclined to select an action from a set of optimal actions as the learning process progresses.

Finally, as more prototype agents are being included in the system and the prototype agents are better attuned to the individual agents, and because the softmax action selection strategy ensures introducing a certain degree of variability in the simulated activity-travel behaviour of agents assigned to the same prototype agent, it is no longer required to copy the prior knowledge of prototype agents gathered in the course of the initial training phase to the matching individual agents and to subsequently refine the agent's individual knowledge. Consequently, each individual agent no longer includes an individually attuned reinforcement learning system and his actions are predicted based on the reinforcement learner incorporated within the corresponding prototype agent. This alteration implies that changes in the reinforcement learning system of the prototype agent impact the decisions of all individual agents assigned to this prototype.

## 5.4 Validation

This section gives an overview of the potential of the algorithm in simulating activity-travel diaries for a large-scale population and at giving an indication of the predictive power of the algorithm.

### 5.4.1 Computational requirements

Firstly, a simulation is run to compare the distribution of the execution time of the system. To this end, an activity-travel pattern on a working day is generated for the individual agent described in section 5.2.2 and assigned membership degrees 0.1782, 0.7414 and 0.0805 for cluster 1, cluster 2 and cluster 3 respectively. As is the case in the previous paragraphs, in this experiment the time resolution for the output sequences equals 5 minutes, while the time

resolution for training the prototype agents equals 15 minutes, and the prototype agent is trained during 15,000 episodes. The overall execution time of the system increases somewhat to 238 seconds. The breakdown of this execution time into its major functions is shown in table 5.12 and is similar to the one displayed in table 5.11.

Function Name	%Time Usage Function+Children
Main	100.00
· Simulation::run	· 98.72
· · Agent::updateQ	· · 86.51
· · · { Module1::updateQ	· · · { 57.87
· · · { Module2::updateQ	· · · { 14.90
· · · { Module3-5::updateQ	· · · { 2.45
· · · · Tree::updateTree	· · · · 70.50
· · · · · Node::updateNode	· · · · · 36.05
· · · · · Node::updateStats	· · · · · 26.81
· · · · · <i>Rest</i>	· · · · · 9.24
· · · · · Node::fitNode	· · · · · 23.21
· · · · · Node::calculateStats	· · · · · 8.67
· · · · · <i>Rest</i>	· · · · · 14.54
· · · · · <i>Rest</i>	· · · · · 11.24
· · · · · <i>Rest</i>	· · · · · 4.72
· · · Agent::calculateRewardmodule3	· · · 7.72
· · · · Inputsequences::calculateSam	· · · · 7.21
· · · · Sequence::calculateUnisam	· · · · 7.08
· · · · · <i>Rest</i>	· · · · · 0.13
· · · · <i>Rest</i>	· · · · 0.51
· · · <i>Rest</i>	· · · 3.57
· · Agent::selectAction	· · 11.92
· · · { Module1::selectAction	· · · { 10.25
· · · { Module2-5::selectAction	· · · { 1.51
· · · <i>Rest</i>	· · · 0.16
· · <i>Rest</i>	· · 0.29
· <i>Rest</i>	· 1.28

Table 5.12: Time usage of functions for the multi-actor reinforcement learning scheduler including a regression tree function approximator incorporating the suggested enhancements

Next, the proposed multi-actor reinforcement learning framework is used to simulate an activity-travel pattern for each of the 148 test cases. To start the analysis, membership degrees are assigned to the test cases based on the tree visualized in table 5.7. The results - displayed in table 5.13 - demonstrate that the test cases are scattered over 10 prototypes based on these membership degrees. Note that not every prototype corresponds to a leaf of

the decision tree: this is due to missing attributes for some cases in the test set. If a case contains a missing value for one of the decision variables included in the nodes of the decision tree, this case is not discarded. Instead, the case is dropped down the tree until either a leaf is reached or until a node is reached for which the attribute is missing. In the latter case, the relative frequencies in this node are used for prediction. As such, the membership degrees of the instances belonging to prototypes 3, 4 and 9 are determined.

Prototype	Membership degrees (in %)			Number of test cases
	Cluster 1	Cluster 2	Cluster 3	
P1	86.96	3.26	9.78	25
P2	83.78	2.70	13.51	26
P3	76.98	2.88	20.14	1
P4	75.76	7.07	17.17	1
P5	71.43	0.00	28.57	4
P6	60.00	26.67	13.33	8
P7	58.70	0.00	41.30	17
P8	45.45	22.73	31.82	7
P9	22.39	67.66	9.95	4
P10	17.82	74.14	8.05	55

Table 5.13: Membership degrees for set of test cases

These prototype agents and their corresponding membership degrees serve as input for the suggested reinforcement learning system to create a one-day activity-travel pattern for each of the 148 test agents. Given this setting, the system runs for 3,663 seconds or approximately 1 hour: 3,662 seconds are required to initialize and train the 10 prototype agents and, after that, 1 second is required to generate the 148 individual activity-travel sequences (i.e. on average 7.2 milliseconds for each individual agent). As the ultimate goal of this research is to simulate the activity-travel behaviour for each individual of a synthetic population covering the research area, as discussed in the introduction to this chapter, and for a time resolution of 5 minutes, these results indicate that - after training the prototype agents - the algorithm would take up approximately 10 hours in order to create 5,000,000 individual activity-travel patterns for one day.

The amount of time required for the initial training phase depends on the number of prototypes and - obviously - on the number of training episodes. However, as a result of the current implementation of the algorithm, once initialized, the prototype agents do not require

retraining as long as the founding activity-travel behaviour does not alter. In addition, even when changes external to the algorithm are assumed to impact the activity-travel behaviour of individuals, the prototype agents do not have to be retrained from scratch. If necessary, the algorithm requires to be retrained in the course of only a limited number of episodes to update its underlying  $Q$ -trees and adapt its behaviour accordingly. Moreover, as already mentioned previously, it is possible to train the prototype agents at a different resolution from the resolution requested for the output sequences, because the regression tree function approximator supporting the reinforcement learning technique enables joining values of state and action variables - including time-related attributes -, which can be treated similarly.

Furthermore, the selected time resolution impacts the timing to a large extent. For instance, for time slots of 15 minutes for the output sequences and of 30 minutes for the training phase, the initialization of the prototype agents lasts for 2,821 seconds (i.e. approximately 47 minutes) and generating a one-day activity-travel sequence for all of the 148 test agents takes 0.36 seconds (i.e. on average 2.4 milliseconds for each individual agent). Finally, the processing time can be substantially reduced by using parallel processors. The current implementation processes all agents sequentially in one time slot before continuing to the next one. Yet it is also feasible to have all agents select the actions to be executed in one time step in parallel and then continue to the next time step. As such, the time required to simulate the daily activity-travel patterns of the synthetic population is distributed among the available processors, decreasing the overall execution time of the algorithm.

Regarding the memory requirements of the current implementation of the algorithm, analyses prove to be bounded by the number of prototype agents, rather than by the number of individual (synthetic) agents. To illustrate this, the algorithm proves to be able to train five prototype agents and generate in total a maximum of 100 individual activity-travel diaries at once, given the current memory restrictions, while it is feasible to generate activity-travel sequences of at least one million individual agents based on only one prototype agent.

#### 5.4.2 Predictive Power

For the purpose of assessing the impact of the suggested improvements on the predictive power of the algorithm, an activity-travel diary is generated for each of the test agents,



for four different scenarios based on the use of the softmax action selection strategy and on the number of prototypes and their corresponding membership degrees. Concerning the softmax action selection strategy, it can be decided to disregard this strategy and apply the  $\epsilon$ -greedy action selection strategy instead. With respect to the definition of the prototypes, the prototypes can be scaled back to match the clusters by reducing the number of agents to the number of clusters and assigning crisp membership degrees (i.e. for a certain prototype, the membership degree of cluster  $x$  equals one if the prototype refers to cluster  $x$ , whereas the membership degrees of the remaining clusters are zero).

The first scenario is the equivalent of the algorithm described in the previous chapter with no softmax action selection and only three prototype agents, each corresponding to one cluster. The second scenario does not use the softmax action selection strategy either; yet this scenario does include the ten prototype agents whose membership degrees are defined in table 5.13. The third and fourth scenarios do apply softmax action selection; in the former scenario, the analyses are conducted based on the three agents matching the three clusters, while in the latter scenario the ten suggested prototype agents are resumed.

Table 5.14 compares the outcome of these four scenarios (15,000 training episodes and a time resolution of 15 minutes for the training phase and 5 minutes during the prediction phase). This table shows that the softmax action selection strategy offers the best opportunity of enhancing the predictive power of the algorithm. The improvement linked to the introduction of a larger number of prototype agents by incorporating the membership degrees for each cluster into the reward functions, is rather limited. The remainder of this section elaborates on the results of the last scenario.

Tables 5.15 and 5.16 provide more insight into the breakdown of the results for each prototype agent. These tables - which are similar to tables 4.22 and 4.23 on page 162 - suggest that, especially based on the sequence alignment method (SAM) comparing the sequences in the long format, this optimized method is particularly suited to simulate individual activity-travel behaviour. Yet, the outcome for prototype  $P10$  is not very favourable. This is caused by the fact that this prototype still covers a fairly large range of individual types of behaviour. The results can be further improved by refining the socio-demographic profiles to segregate these underlying types of behaviour. This can be realized by relaxing the constraints guiding

<b>Softmax action selection</b>	<b>No</b>	<b>No</b>	<b>Yes</b>	<b>Yes</b>
<b>Number of prototypes</b>	<b>3</b>	<b>10</b>	<b>3</b>	<b>10</b>
SAM in set of test sequences				
Avg.	0.34	0.34	0.34	0.34
St.dev.	0.13	0.13	0.13	0.13
SAM in set of simulated sequences				
Avg.	0.39	0.13	0.35	0.35
St.dev.	0.12	0.05	0.16	0.13
SAM between test sequences and simulated sequences				
Avg.	0.40	0.36	0.36	0.35
St.dev.	0.09	0.06	0.12	0.12
t-statistic	7.97	3.71	1.29	1.18
P-value	0.0000	0.0001	0.0989	0.1197

Table 5.14: Validation results for DP-SAM calculated based on the long format sequences simulated by the multi-actor reinforcement learning system for four scenarios

the construction of the decision tree, for instance by reducing the minimum number of observations required in each node or by decreasing the threshold value related to the splitting criterion.

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	All P
SAM in set of test sequences											
Avg.	1.07	1.25	NA	NA	0.96	1.70	1.57	1.41	1.34	1.43	<b>1.39</b>
St.dev.	0.55	0.88	NA	NA	0.27	0.52	0.57	0.56	0.80	0.59	<b>0.60</b>
SAM in set of simulated sequences											
Avg.	1.45	1.05	NA	NA	1.13	1.41	1.80	1.10	1.04	1.90	<b>1.84</b>
St.dev.	0.54	0.88	NA	NA	0.29	0.48	0.56	0.56	0.97	0.59	<b>0.60</b>
SAM between test sequences and simulated sequences											
Avg.	1.38	1.32	1.40	1.50	1.65	1.43	1.86	1.57	2.36	1.97	<b>1.69</b>
St.dev.	0.36	0.55	NA	NA	0.30	0.58	0.50	0.33	0.16	0.41	<b>0.52</b>
t-statistic	3.72	0.58	NA	NA	1.96	-1.21	2.17	1.20	8.20	9.64	<b>6.82</b>
P-value	0.0003	0.2830	NA	NA	0.0456	0.8750	0.0199	0.1332	0.0000	0.0000	<b>0.0000</b>

Table 5.15: Validation results for DP-SAM calculated based on the short format sequences simulated by the optimized multi-actor reinforcement learning system

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	All P
SAM in set of test sequences											
Avg.	0.27	0.23	NA	NA	0.19	0.38	0.34	0.38	0.44	0.36	<b>0.34</b>
St.dev.	0.13	0.11	NA	NA	0.10	0.10	0.15	0.10	0.08	0.11	<b>0.13</b>
SAM in set of simulated sequences											
Avg.	0.22	0.14	NA	NA	0.25	0.20	0.28	0.20	0.11	0.31	<b>0.35</b>
St.dev.	0.10	0.11	NA	NA	0.08	0.08	0.11	0.09	0.07	0.10	<b>0.13</b>
SAM between test sequences and simulated sequences											
Avg.	0.29	0.23	0.33	0.22	0.30	0.38	0.39	0.38	0.52	0.42	<b>0.35</b>
St.dev.	0.10	0.10	NA	NA	0.17	0.13	0.11	0.10	0.05	0.08	<b>0.12</b>
t-statistic	0.95	0.04	NA	NA	1.23	-0.01	1.53	0.22	2.11	5.59	<b>1.18</b>
P-value	0.1749	0.4858	NA	NA	0.1390	0.5049	0.0699	0.4151	0.0341	0.0000	<b>0.1197</b>

Table 5.16: Validation results for DP-SAM calculated based on the long format sequences simulated by the optimized multi-actor reinforcement learning system

Tables 5.17 to 5.26 display some descriptive statistics for both the test and simulated sequences. For the majority of the prototypes, the average number of activities in the simulated sequences exceeds the average length of the corresponding test sequences. Next, table 5.18 shows that sleep-home-sleep sequences are not predicted by the reinforcement learning scheduler, while such sequences are indeed observed in the set of training sequences (51% of the sequences of cluster 1 and 67% of the sequences in cluster 3 are sleep-home-sleep sequences (cf. table 5.2)). This observation can be attributed to the fact that the prediction of the duration is attuned to the duration boundaries (as explained in section 4.2) of the activity episodes observed in the training set. As a result, for each prototype agent the simulated duration approaches the average duration of the compounding clusters, according to the defined membership degrees. In case of the in-home activities, these average durations equal 507, 209 and 443 for cluster 1, 2 and 3 respectively. These durations are predicted fairly well as recorded in table 5.22. Yet, when comparing the durations of the simulated in-home activities in table 5.22 to the average of the observed durations in the test sequences in table 5.21, it attracts the attention that the average duration of the simulated in-home activities is often considerably smaller than the average duration of the in-home activities observed in the test set. Consequently, in order to complete the schedule for the day (i.e. to cover 1440 minutes), the reinforcement learning scheduler adds an out-of-home activity. Because of this, the sleep-home-sleep sequence is not simulated at all.

With respect to the non in-home activities, tables 5.21 and 5.22 indicate that the simulated durations approach the observed durations better. Furthermore, the averages of the total duration of an activity within a sequence of the test and simulated sequences - displayed in tables 5.23 and 5.24 respectively - substantiate the favourable results presented in table 5.16.

Concerning the activity locations, tables 5.25 and 5.26 show that the majority of the sequences generated for prototypes 9 and 10 mainly contain a large radius of action (25-50km and  $> 50$ km), which is clearly founded on the distribution of the locations of cluster 2 (cf. table 5.6), matching the membership degrees defined in table 5.13. For prototypes 1 to 8, which principally correspond to cluster 1 according to their membership degrees, table 5.26 displays a larger spread of the locations over the distance bands but with an inclination

towards the smaller distance bands, as is the case for cluster 1 (cf. table 5.6). For some prototypes, these results are less clear-cut, as they do not contain many observations.

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
#seq	25	26	1	1	4	8	17	7	4	55
Avg.#act	4.04	4.65	3.00	5.00	4.00	5.38	4.24	4.71	4.00	4.89
% of sequences which match the following patterns:										
S-H-S	52.00	42.31	100.00	0.00	50.00	25.00	41.18	28.57	50.00	27.27
S-H-{W}-H-S	0.00	7.69	0.00	0.00	0.00	25.00	0.00	0.00	25.00	45.45
S-H-{not W}-H-S	40.00	42.31	0.00	100.00	50.00	25.00	35.29	42.86	0.00	9.09

Table 5.17: Some general descriptive statistics of the test sequences

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
#seq	25	26	1	1	4	8	17	7	4	55
Avg.#act	5.16	5.08	6.00	5.00	6.00	5.00	4.65	5.00	5.00	5.84
% of sequences which match the following patterns:										
S-H-S	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
S-H-{W}-H-S	24.00	7.69	0.00	0.00	0.00	0.00	0.00	57.14	0.00	7.27
S-H-{not W}-H-S	64.00	76.92	0.00	100.00	0.00	75.00	29.41	42.86	0.00	0.00

Table 5.18: Some general descriptive statistics of the simulated sequences

<b>Prototype</b>	<b>P1</b>	<b>P2</b>	<b>P3</b>	<b>P4</b>	<b>P5</b>	<b>P6</b>	<b>P7</b>	<b>P8</b>	<b>P9</b>	<b>P10</b>
<b>H</b>	96.00	100.00	100.00	100.00	100.00	100.00	88.24	100.00	100.00	98.18
<b>S</b>	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
<b>W</b>	0.00	7.69	0.00	0.00	0.00	50.00	5.88	0.00	50.00	61.82
<b>M</b>	8.00	26.92	0.00	100.00	50.00	25.00	23.53	71.43	25.00	25.45
<b>D</b>	40.00	34.62	0.00	0.00	0.00	50.00	47.06	28.57	0.00	25.45

Table 5.19: Percentage of test sequences containing the listed activities at least once

<b>Prototype</b>	<b>P1</b>	<b>P2</b>	<b>P3</b>	<b>P4</b>	<b>P5</b>	<b>P6</b>	<b>P7</b>	<b>P8</b>	<b>P9</b>	<b>P10</b>
<b>H</b>	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
<b>S</b>	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
<b>W</b>	32.00	15.38	0.00	0.00	25.00	12.50	23.53	57.14	100.00	100.00
<b>M</b>	12.00	15.38	0.00	0.00	0.00	50.00	35.29	0.00	0.00	38.18
<b>D</b>	68.00	80.77	100.00	100.00	75.00	37.5	76.47	42.86	25.00	54.55

Table 5.20: Percentage of simulated sequences containing the listed activities at least once

	P1			P2			P3			P4			P5		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	37	544	389	46	445	324	1	780	0	2	428	435	6	518	371
<b>S</b>	51	443	124	53	461	102	2	429	19	2	484	58	8	442	168
<b>W</b>	0	-	-	2	245	21	0	-	-	0	-	-	0	-	-
<b>M</b>	2	28	11	11	37	28	0	-	-	1	60	0	2	165	184
<b>D</b>	11	206	154	9	143	88	0	-	-	0	-	-	0	-	-

	P6			P7			P8			P9			P10		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	14	233	233	24	476	359	10	454	389	5	435	376	92	281	244
<b>S</b>	16	453	111	34	469	64	14	446	28	8	447	13	110	437	89
<b>W</b>	6	370	142	1	780	0	0	-	-	2	600	64	39	446	192
<b>M</b>	3	53	32	5	12	6	7	253	157	1	50	0	14	58	83
<b>D</b>	4	144	132	8	268	235	2	55	7	0	-	-	14	101	97

Table 5.21: Number of activity episodes and average and standard deviation of the duration of these activity episodes in the test sequences



	P1			P2			P3			P4			P5		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	49	427	95	53	427	92	3	327	244	2	445	85	12	306	156
<b>S</b>	50	465	1	49	465	1	2	465	0	2	465	0	8	465	0
<b>W</b>	8	232	21	4	270	0	0	-	-	0	-	-	1	290	0
<b>M</b>	4	101	2	4	100	0	0	-	-	0	-	-	0	-	-
<b>D</b>	18	141	2	22	150	2	1	210	0	1	140	0	3	265	5

	P6			P7			P8			P9			P10		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	17	436	90	30	442	104	14	366	137	7	252	184	100	244	152
<b>S</b>	15	460	0	26	465	0	14	450	0	4	450	0	96	437	59
<b>W</b>	1	335	0	4	254	93	4	340	0	8	383	78	71	423	82
<b>M</b>	4	90	0	6	129	7	0	-	-	0	-	-	23	81	17
<b>D</b>	3	160	0	13	275	70	3	270	0	1	10	0	31	115	21

Table 5.22: Number of activity episodes and average and standard deviation of the duration of these activity episodes in the simulated sequences

	P1			P2			P3			P4			P5		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	24	838	246	26	788	178	1	780	0	1	855	0	4	776	182
<b>S</b>	25	903	195	26	940	168	1	857	0	1	967	0	4	884	262
<b>W</b>	0	-	-	2	245	21	0	-	-	0	-	-	0	-	-
<b>M</b>	2	28	11	7	58	41	0	-	-	1	60	0	2	165	184
<b>D</b>	10	227	147	9	143	88	0	-	-	0	-	-	0	-	-

	P6			P7			P8			P9			P10		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	8	408	264	15	761	383	7	649	311	4	544	350	54	479	262
<b>S</b>	8	907	179	17	937	89	7	892	33	4	894	19	55	873	133
<b>W</b>	4	555	50	1	780	0	0	-	-	2	600	64	34	511	165
<b>M</b>	2	80	64	4	15	4	5	354	221	1	50	0	14	58	83
<b>D</b>	4	144	132	8	268	235	2	55	7	0	-	-	14	101	97

Table 5.23: Number of test sequences containing activities and average and standard deviation of the total duration of the activities in these sequences

	P1			P2			P3			P4			P5		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	25	836	98	26	870	50	1	980	0	1	890	0	4	919	13
<b>S</b>	25	930	2	26	860	168	1	470	0	1	930	0	4	470	0
<b>W</b>	8	232	21	4	270	0	0	-	-	0	-	-	1	290	0
<b>M</b>	3	135	61	4	100	0	0	-	-	0	-	-	0	-	-
<b>D</b>	17	149	35	21	157	33	1	210	0	1	140	0	3	265	5

	P6			P7			P8			P9			P10		
	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.	#	Avg.	Sd.
<b>H</b>	8	927	80	17	781	182	7	731	42	4	441	28	55	443	50
<b>S</b>	8	806	212	17	711	239	7	900	0	4	450	0	55	769	200
<b>W</b>	1	335	0	4	254	93	4	340	0	4	766	22	55	546	130
<b>M</b>	4	90	0	6	129	7	0	-	-	0	-	-	21	89	30
<b>D</b>	3	160	0	13	275	70	3	270	0	1	10	0	30	119	25

Table 5.24: Number of simulated sequences containing activities and average and standard deviation of the total duration of the activities in these sequences

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
<b>Home</b>	56.00	42.31	100.00	0.00	50.00	25.00	41.18	28.57	50.00	27.27
<b>&lt;2km</b>	12.00	11.54	0.00	0.00	0.00	0.00	11.76	0.00	0.00	5.45
<b>2-5km</b>	8.00	3.85	0.00	100.00	0.00	12.50	5.88	14.29	0.00	10.91
<b>5-10km</b>	12.00	19.23	0.00	0.00	50.00	12.50	5.88	28.57	0.00	12.73
<b>10-25km</b>	8.00	15.38	0.00	0.00	0.00	25.00	0.00	0.00	50.00	12.73
<b>25-50km</b>	0.00	7.69	0.00	0.00	0.00	0.00	11.76	14.29	0.00	12.73
<b>&gt;50km</b>	4.00	0.00	0.00	0.00	0.00	25.00	23.53	14.29	0.00	18.18

Table 5.25: Percentage of test sequences in which the listed distance bands corresponds to the farthest location reached

Prototype	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
<b>Home</b>	8.00	65.38	0.00	0.00	25.00	12.50	0.00	0.00	0.00	0.00
<b>&lt;2km</b>	16.00	0.00	0.00	0.00	0.00	25.00	47.06	0.00	0.00	3.64
<b>2-5km</b>	24.00	11.54	0.00	100.00	0.00	12.50	23.53	0.00	0.00	3.64
<b>5-10km</b>	4.00	3.85	0.00	0.00	0.00	0.00	17.65	0.00	0.00	1.82
<b>10-25km</b>	4.00	0.00	0.00	0.00	0.00	37.50	5.88	14.29	0.00	3.64
<b>25-50km</b>	44.00	15.38	100.00	0.00	50.00	12.50	5.88	85.71	75.00	74.55
<b>&gt;50km</b>	0.00	3.85	0.00	0.00	25.00	0.00	0.00	0.00	25.00	12.73

Table 5.26: Percentage of simulated sequences in which the listed distance bands corresponds to the farthest location reached

To illustrate the effect of refining the decision tree containing the socio-demographic profiles, the branch of the decision tree (presented in table 5.7) containing the cases of prototype *P10* - which corresponds to node number 10 - is refined by relaxing the threshold value of the splitting criterion used in fitting the decision tree (in this case the minimum required ratio of the within-node deviance of the node to be split compared to the root node is reduced from 0.01 to 0.005). The additional refinement of this branch is shown in table 5.27. Table 5.28 displays the new membership distribution of the 55 affected test cases.

Test rule	NodeNr.	#	Pred.	%clu1	%clu2	%clu3	Leaf
<i>Workinghours</i> > 20.5	10	174	2	0.0695	0.2892	0.0314	
<i>Age</i> < 37.5	13	68	2	0.0314	0.0035	0.0000	
<i>Weekday</i> : <i>M, T, D</i>	15	40	2	0.0045	0.0041	0.0000	
<i>#Child</i> : 0	17	14	2	0.0045	0.0038	0.0000	*
<i>#Child</i> : 1, 2, 3	18	26	2	0.0000	0.0043	0.0000	*
<i>Weekday</i> : <i>W, F</i>	16	28	2	0.0269	0.0026	0.0000	
<i>Gender</i> : <i>m</i>	19	20	2	0.0135	0.0031	0.0000	*
<i>Gender</i> : <i>f</i>	20	8	1	0.0135	0.0006	0.0000	*
<i>Age</i> > 37.5	14	106	2	0.0381	0.0032	0.0000	
<i>Age</i> < 46.5	21	62	2	0.0179	0.0030	0.0001	
<i>#Child</i> : 0, 1, 2	23	44	2	0.0135	0.0026	0.0001	
<i>#Child</i> : 0, 1	25	13	3	0.0090	0.0021	0.0003	*
<i>#Child</i> : 2	26	31	2	0.0045	0.0032	0.0000	
<i>Age</i> < 43.5	27	26	2	0.0000	0.0034	0.0000	
<i>Workinghours</i> < 39.5	29	17	2	0.0000	0.0029	0.0000	
<i>Work</i> : <i>p</i>	31	5	2	0.0000	0.0045	0.0000	*
<i>Work</i> : <i>f</i>	32	12	2	0.0000	0.0022	0.0000	
<i>WD</i> : <i>M, F</i>	33	6	2	0.0000	0.0037	0.0000	*
<i>WD</i> : <i>T, W, D</i>	34	6	3	0.0000	0.0011	0.0000	*
<i>Workinghours</i> > 39.5	30	9	2	0.0000	0.0045	0.0000	*
<i>Age</i> > 43.5	28	5	1	0.0045	0.0009	0.0002	*
<i>#Child</i> : 3, 3+	24	18	2	0.0045	0.0040	0.0000	
<i>Age</i> < 42.5	35	12	2	0.0000	0.0045	0.0000	*
<i>Age</i> > 42.5	36	6	2	0.0045	0.0030	0.0000	*
<i>Age</i> > 46.5	22	44	2	0.0202	0.0035	0.0000	
<i>Weekday</i> : <i>M, T, W, D</i>	37	35	2	0.0135	0.0037	0.0000	*
<i>Weekday</i> : <i>F</i>	38	9	2	0.0067	0.0025	0.0001	*

Table 5.27: Refined socio-demographical profiles of prototype P10

---

Prototype	Membership degrees (in %)			Number of test cases
	Cluster 1	Cluster 2	Cluster 3	
R1	40.00	40.00	20.00	4
R2	33.33	66.67	0.00	5
R3	33.33	55.56	11.11	4
R4	30.77	30.77	38.46	7
R5	30.00	70.00	0.00	6
R6	17.14	82.86	0.00	13
R7	0.00	100.00	0.00	4
R8	0.00	96.15	3.85	6
R9	0.00	83.33	16.67	1
R10	0.00	16.67	83.33	5

---

Table 5.28: Refined membership degrees for set of test cases matching prototype P10

Next, 10 additional prototype agents are fitted and used to generate activity-travel sequences, in order to replace the 55 simulated individuals assigned to prototype P10 in the previous analyses. The outcome of this simulation, which is presented in tables 5.29 and 5.30, reveals that refining the socio-demographic profiles matching prototype *P10* improves the resemblance between the simulated activity-travel patterns and the observed test cases.

Prototype	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	All R
SAM in set of test sequences											
Avg.	0.72	1.52	1.33	1.41	1.38	1.26	0.85	1.57	NA	1.71	<b>1.42</b>
St.dev.	0.79	0.42	0.26	0.47	0.52	0.43	0.34	0.36	NA	0.52	<b>0.51</b>
SAM in set of simulated sequences											
Avg.	2.12	1.42	2.37	1.74	1.28	1.58	1.43	1.62	NA	1.90	<b>1.77</b>
St.dev.	0.54	0.61	0.64	0.68	0.38	0.55	0.51	0.49	NA	0.46	<b>0.56</b>
SAM between test sequences and simulated sequences											
Avg.	1.71	1.65	1.85	1.54	1.47	1.58	2.00	1.80	1.60	1.99	<b>1.69</b>
St.dev.	0.55	0.57	0.57	0.57	0.53	0.36	0.43	0.40	NA	0.37	<b>0.46</b>
t-statistic	2.33	0.42	1.68	0.53	0.33	2.95	4.46	1.21	NA	1.16	<b>4.15</b>
P-value	0.0243	0.3433	0.0856	0.3029	0.3742	0.0042	0.0026	0.1300	NA	0.1357	<b>0.0001</b>

Table 5.29: Validation results for DP-SAM calculated based on the short format sequences simulated by the optimized multi-actor reinforcement learning system, refined for prototype P10

Prototype	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	All R
SAM in set of test sequences											
Avg.	0.25	0.30	0.38	0.31	0.36	0.34	0.29	0.41	NA	0.31	<b>0.36</b>
St.dev.	0.18	0.18	0.05	0.13	0.12	0.09	0.15	0.10	NA	0.07	<b>0.11</b>
SAM in set of simulated sequences											
Avg.	0.37	0.25	0.39	0.34	0.21	0.29	0.26	0.26	NA	0.32	<b>0.36</b>
St.dev.	0.07	0.13	0.09	0.12	0.11	0.10	0.08	0.08	NA	0.13	<b>0.11</b>
SAM between test sequences and simulated sequences											
Avg.	0.42	0.29	0.43	0.40	0.38	0.41	0.42	0.45	0.41	0.41	<b>0.40</b>
St.dev.	0.08	0.09	0.08	0.08	0.11	0.09	0.01	0.06	NA	0.05	<b>0.08</b>
t-statistic	2.04	-0.19	1.09	2.17	0.35	2.65	2.08	1.02	NA	3.17	<b>3.91</b>
P-value	0.0397	0.5735	0.1651	0.0219	0.3650	0.0088	0.0450	0.1628	NA	0.0048	<b>0.0001</b>

Table 5.30: Validation results for DP-SAM calculated based on the long format sequences simulated by the optimized multi-actor reinforcement learning system, refined for prototype P10



## 5.5 Conclusions

To conclude, the current chapter focuses on the scalability and validity of the presented framework incorporating a multi-actor reinforcement learning algorithm amended with a regression tree function approximator in simulating activity-travel sequences in an activity-based travel-demand model. Therefore, this chapter walks through the entire prediction process. With respect to the scalability, the execution time of the algorithm is analysed and two major methodological suggestions to enhance the performance of the algorithm are advanced.

The first suggestion proposes to increase the number of prototype agents. To this end, the one-to-one relation between the prototype agents and the clusters is broken. Instead, the design of prototype agents is now founded on the membership degrees for each cluster, estimated in the nodes and leaves of the classification tree determining the socio-demographic profiles, and the reward functions incorporated in the reinforcement learning system are adapted accordingly.

Secondly, this chapter suggests to include a softmax action selection strategy rather than an  $\epsilon$ -greedy action selection strategy. This action selection strategy allows the agent to estimate a probability distribution for each set of actions consistent with the corresponding  $Q$ -values. As a result of implementing both suggestions, it is no longer required to copy the prior knowledge of the prototype agents - which is built up in the initial training phase - to the individual agents, and to further refine each of these individual agents' knowledges.

The outcome of the analyses conducted in this chapter, indicates that the framework designed here is capable of accurately simulating individual activity-travel behaviour based on observed activity-travel diaries. Furthermore, the proposed system is able to do so, while requiring acceptable memory capacity and within a realistic time frame, even when aiming at generating activity-travel diaries for a large population (e.g. more than 5 million agents). Additionally, once the prototype agents are initialized, they do not have to be retrained for further analysis as long as the underlying behaviour does not change. If this behaviour does alter, the prototype agents only require to be updated rather than to be retrained from scratch as they are able to recover previously gathered knowledge.



## Chapter 6

# Final Conclusions

### 6.1 Key Findings

The research presented in this manuscript aims at settling the following main research problem:

*Is reinforcement learning able to constitute a solid basis for modelling individual activity-travel behaviour?*

In order to provide a comprehensive answer to this research question, a number of sub-questions are defined:

1. (a) What are the aspects of reinforcement learning restricting its applicability in the current study area?  
(b) Which adaptations are required to meet these limitations?
2. To which extent is reinforcement learning able to account for interactions?
3. (a) Which type of data is required to serve as input?  
(b) To which extent do these data require pre-processing for the benefit of the reinforcement learning algorithm?
4. Can a conceptual framework incorporating reinforcement learning be defined which aims at simulating activity-travel sequences?

5. (a) To which extent is this reinforcement learning framework able to generate meaningful activity-travel sequences based on observed data?
- (b) To which extent is this reinforcement learning framework able to do so within an acceptable time frame for a given synthetic population?

The remainder of this section attempts to formulate an answer to each of these subquestions.

**What are the aspects of reinforcement learning restricting its applicability in the current study area?**

In the current research, a well-known variant of reinforcement learning labelled  $Q$ -learning is applied. The advantage of this approach includes the fact that no model of the environment is required when learning the optimal policy. The rewards guiding the reinforcement learning approach are processed in so-called  $Q$ -values. Traditionally, these  $Q$ -values are stored in a table consisting of one entry for each feasible (state,action)-pair. Yet, for problems containing large state and action spaces, this  $Q$ -learning approach is subject to the curse of dimensionality, forcing both the memory requirements and the computational time to rise enormously (Sutton & Barto, 1998).

Furthermore, linked to the previous restriction, the  $Q$ -learning approach requires gathering information on all feasible (state,action)-pairs by visiting each at least once and preferably an infinite number of times, which is impracticable in large state and action spaces (Sutton & Barto, 1998).

**Which adaptations are required to meet these limitations?**

To tackle these issues, the present research introduces a function approximator which allows the reinforcement learning agent to generalize the state and action spaces simultaneously. To this end, a regression tree induction algorithm - described by Potts & Sammut (2005) - is advanced, which is able to process a numeric dependent variable, to handle large streams of on-line data and to be responsive to structural changes in these continuous streams of data. The applicability of this regression tree induction algorithm is demonstrated by means of a case study.

**To which extent is reinforcement learning able to account for interactions?**

Traditional  $Q$ -learning is particularly suited to account for delayed rewards, but it cannot take into consideration interactions between a number reinforcement learning systems. But, activity-based travel-demand models have to be able to counter interactions between individuals, in particular household members, on the one hand (Timmermans, 2006), and between the various attributes of activity-travel sequences on the other hand (Gärling *et al.*, 1997; Joh *et al.*, 2002). For this purpose, the concepts of multi-actor reinforcement learning are presented. The relevance of this technique is illustrated based on a case study drawn from the current research area.

**Which type of data is required to serve as input?**

For the purpose of calibrating the parameters of the system and of the reward functions, the data applied here originate from observed activity-travel diaries and contain the activity type, activity duration and timing, activity location and travel mode used to get to this location. Moreover, before being incorporated in the scheduling engine, these data are linked to observed socio-demographic data, as is indicated in the next paragraph.

**To which extent do these data require pre-processing for the benefit of the reinforcement learning algorithm?**

Because the observed activity-travel sequences differ due to variances in individual needs, preferences, opportunities and constraints, the observed activity-travel sequences are split into groups of sequences which are assumed to display similar activity-travel behaviour. To this end, a method introduced by Wilson (2008) can be applied, which calculates the dissimilarity of sequences based on the well-known sequence alignment method (Wilson, 1998a) while taking the geographic location of the activities into account. However, Wilson's algorithm disregards the relative position of the locations within a sequence and the distances travelled within this sequence. Therefore, the current research proposes a spatio-temporal dissimilarity measure. Applying this method, a number of groups (clusters) of sequences can be defined. These clusters are matched to observed socio-demographic data and subsequently, for each of these clusters a socio-demographic profile is constructed by means of

a classification tree. The purpose of these socio-demographic profiles consists of splitting the synthetic population into more homogeneous groups with respect to their activity-travel behaviour.

**Can a conceptual framework incorporating reinforcement learning be defined aiming at simulating activity-travel sequences?**

The scheduling framework constituting the core of this research contains five decision modules. Each of these five decision modules includes a reinforcement learning system incorporating a regression tree function approximator and is held responsible for simulating one particular aspect of activity-travel patterns. The first module determines the duration of the activity, the second decides whether or not to execute a fixed activity (i.e. sleeping or working), the third concentrates on choosing the activity to be performed, the fourth selects the activity location and the fifth focuses on the travel mode to be used. The decisions taken in the course of this scheduling process are linked by applying multi-actor reinforcement learning.

Furthermore, a number of prototype agents - each of which matches one of the clusters distinguished in the course of the data pre-processing phase - are defined. These prototype agents are initialized and trained first to be able to provide prior knowledge to the agents, corresponding to a member of the synthetic population. The analyses in this manuscript show that this framework can be calibrated based on the available observed data.

**To which extent is this reinforcement learning framework able to generate meaningful activity-travel sequences based on observed data?**

The resulting activity-travel sequences are compared to some test sequences by means of the multidimensional sequence alignment method implemented by Joh *et al.* (2001), and various activity-travel related attributes are analysed by means of some descriptive statistics. The outcomes of these analyses prove that the simulated sequences match the observed activity-travel patterns well. These results are even improved by refining the proposed framework through the incorporation of more prototype agents - each of which corresponds to a node in the classification tree rather than to a cluster - and the introduction of a softmax action

selection strategy.

**To which extent is this reinforcement learning framework able to do so within an acceptable time frame for a given synthetic population?**

The performance of the algorithm is examined with respect to execution time and memory requirements. These analyses demonstrate that the algorithm is capable of simulating the activity-travel behaviour for a realistic synthetic population within an acceptable time frame: after initializing the prototype agents - a process which lasts about 1 hour for 10 prototype agents - approximately 10 hours are required to generate for instance 5 million individual activity-travel patterns with a time resolution of 5 minutes. Obviously, the total execution time is impacted by the selected time resolution and the number of training episodes, as well as by the number of prototype agents. Yet, once initialized, the prototype agents can be re-used because they do not require retraining for each simulation as long as the underlying activity-travel behaviour does not change. Additionally, the simulation process can be accelerated by implementing it on parallel processors which simulate the decisions taken by the agents in each time step simultaneously before continuing to the next time step.

## **6.2 Topics for Further Research**

Despite the favourable results presented in this research, some topics for future research remain. The first topic regards the proposed spatio-temporal dissimilarity measure incorporating the geographical location. Before being able to apply this technique to large databases, its scalability and computational efficiency has to be examined and improved. Furthermore, the possibility of adapting the actual computation of the dissimilarity measure - i.e. after having transformed and normalized the geographical information within sequences - to a more sophisticated multidimensional sequence alignment method as advanced by Joh (2004) should be investigated.

Secondly, Arentze & Timmermans (2003) point out that having previous knowledge of a domain and applying it to take more informed decisions concerning alternative choices could improve the system by searching in a more intelligent way and learn faster. Examples of such prior knowledge include information on travel modes, on land use or on spatio-temporal

constraints (for instance opening hours of shops and public services). To this end, further research is required on the introduction of prior knowledge on the environment and state-action values into the reinforcement learning algorithm. Yet, the latter part is partially already covered by the incorporation of the regression tree, which allows the reinforcement agent to generalize over state-action values.

In addition, - next to accounting for interactions between decision components - Timmermans (2006) indicates the importance of introducing interactions between individuals, in particular between household members, as these interactions co-found the generation of individual activity-travel patterns. From this point of view, Timmermans (2006) states that the majority of the existing activity-based modelling efforts focuses on maximizing individual utility functions, instead of maximizing household utility functions. Therefore, future research efforts have to concentrate on incorporating household interactions as well.

Fourthly, the parameter settings used throughout the entire simulation process demand additional calibration in order to further improve the performance of the algorithm. From this perspective, a more profound investigation of the functional forms of the reward functions guiding the reinforcement learning process is required as well. Moreover, the sensitivity of the reinforcement learning approach to changes either in environmental parameters or to changes in the underlying activity-travel behaviour has to be examined.

Furthermore, the presented framework consists of a chain of processes which control the data pre-processing, calculate dissimilarities between the observed activity-travel patterns, distinguish the clusters of sequences displaying similar activity-travel behaviour and construct socio-demographic profiles which correspond to these clusters, calibrate the parameters for the reinforcement learning system (mainly for the reward functions) based on these analyses, assign membership degrees for each individual in the synthetic population, run the reinforcement learning framework to extract activity-travel sequences for all of these individuals and analyse this outcome. However, the majority of these building blocks are not integrated and the micro simulation process therefore needs a large amount of manual intervention. Consequently, a fifth item of further research includes streamlining this process from data pre-processing to handling the resulting activity-travel patterns for each member of the synthetic population.



Sixthly, the current implementation of the scheduling framework does not determine the exact geographic location of each individual for each time step, as it does not include a geographic network on which the activity-travel patterns can be projected. Future research efforts should focus on integrating the presented scheduling algorithm with an activity-based travel-demand model which holds a geographic network and which contains an algorithm to set the geographic location of an activity.

Last but not least, further research has to examine the conversion of the framework described in this manuscript into a dynamic activity-based travel-demand model in which individuals determine their activity-travel sequences dynamically by entering the transportation network simultaneously and interacting with each other (Arentze *et al.*, 2005). Although the current research only provides a rather static implementation, the reinforcement learning technique founding the scheduling framework submitted here is particularly suited to enable dynamic interaction with the environment and to incorporate learning. Moreover, the presented scheduling algorithm is conceptualized as such that the individual agent schedules as time progresses - as opposed to scheduling ahead -, which allows the system to evaluate the impact of unexpected events in the course of the schedule execution, for instance delays on the transportation network.



## Appendix A

### Sequences in Short Form: Results

This appendix presents the results of estimating the dissimilarity and partitioning the data set into groups of displaying similar activity-travel patterns based on the sequences in the short form as discussed in section 3.4.1 of chapter 3.

#### A.1 Identification of Groups of Similar Behaviour

##### A.1.1 Clustering

The number of clusters  $k$  is set to 6 based on table A.1 and Figure A.1.

The number of clusters for the dissimilarity scores determined by means of the existing technique is also set to 6 as shown in table A.2 and figure A.2.

Tables A.3 and A.4 show the statistics per cluster for the geographical parameters. Founded on these tables and the results of the corresponding ANOVA tests in table A.5, the same conclusions as formulated in section 3.4.1 can be drawn. The absolute geographical location does not distort the ability of the proposed spatio-temporal dissimilarity measure to identify comparable spatial sequences, while the geographical location - rather than the spatial dispersion of the sequences - does impact the outcome of the existing distance algorithm.

---

APPENDIX A. SEQUENCES IN SHORT FORM: RESULTS

---

$k$	Avg sil. width	Min. #	Max. #
2	0.5893	230	364
3	0.5997	134	230
4	0.6403	52	230
5	0.6467	37	230
6	0.6583	37	230
7	0.6437	26	230
8	0.5544	26	230
9	0.5574	12	230
10	0.5607	10	230
11	0.5616	10	230
12	0.5474	10	230
13	0.5505	9	230
14	0.5406	9	230
15	0.5587	9	230
16	0.5495	9	230
17	0.5543	9	219
18	0.5399	9	219
19	0.5522	9	219
20	0.5590	9	219

Table A.1: Average silhouette width for varying number of clusters  $k$  based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3 for the sequences in the short form

$k$	Avg sil. width	Min. #	Max. #
2	0.5181	276	318
3	0.4654	144	238
4	0.4584	54	205
5	0.4323	49	145
6	0.4443	36	145
7	0.3840	36	136
8	0.3737	34	115
9	0.3566	34	113
10	0.3416	34	113
11	0.3501	19	113
12	0.3398	16	112
13	0.3162	10	80
14	0.3150	10	80
15	0.3162	10	80
16	0.3170	10	80
17	0.3196	10	80
18	0.3257	10	80
19	0.3412	10	80
20	0.3343	10	80

Table A.2: Average silhouette width for varying number of clusters  $k$  based on the existing distance measure described in section 3.3.2 for the sequences in the short form

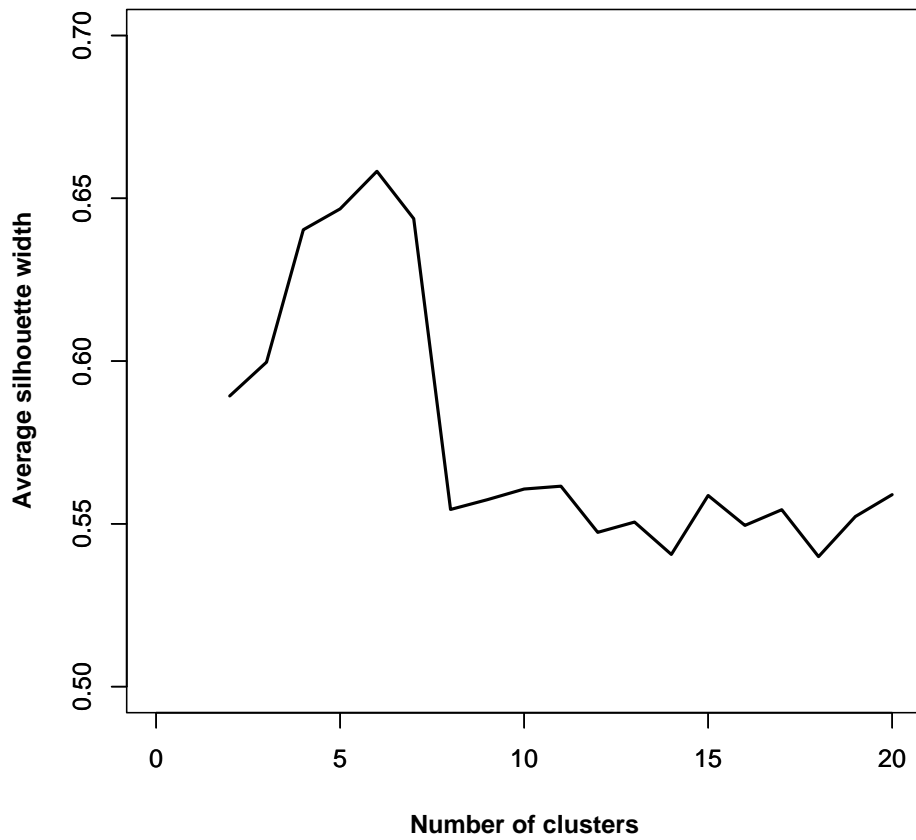


Figure A.1: Cluster results for varying number of clusters  $k$  based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3 for the sequences in the short form

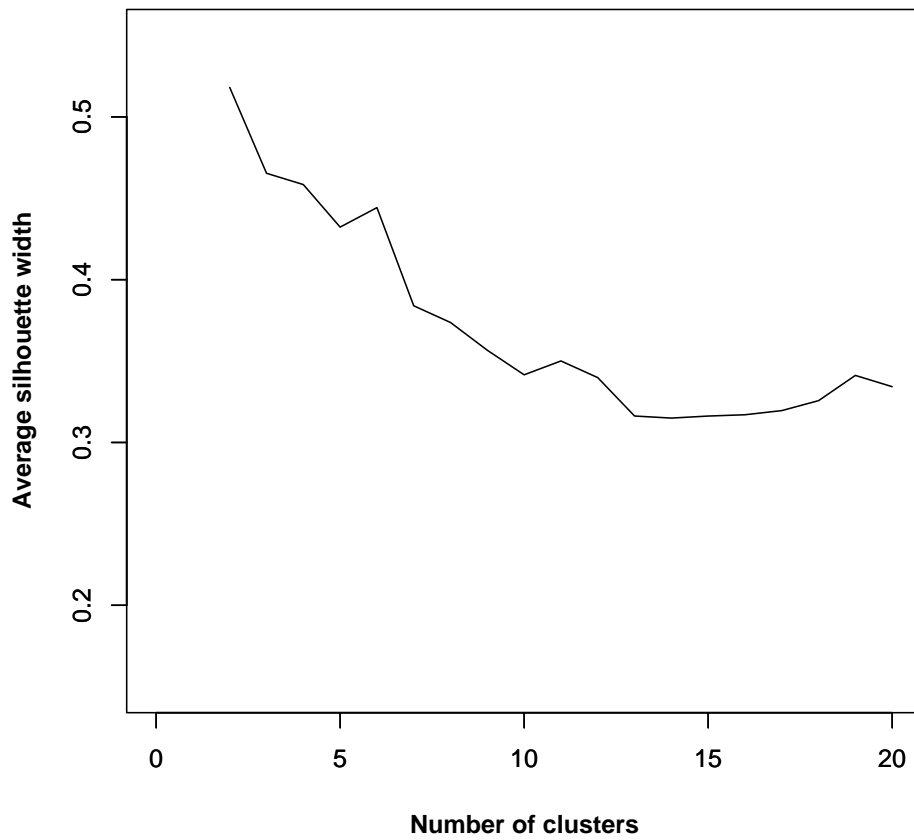


Figure A.2: Cluster results for varying number of clusters  $k$  based on the existing distance measure described in section 3.3.2 for the sequences in the short form

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	All
# of sequences	230	47	149	52	79	37	<b>594</b>
# of episodes per sequence							
Avg	3.00	6.02	5.01	4.00	7.18	9.92	<b>4.82</b>
St. dev.	0.00	0.15	0.12	0.00	0.38	1.99	<b>2.02</b>
Eccentricity							
Avg	0.0478	0.9879	0.9796	1.0000	0.9966	0.9909	<b>0.6242</b>
St. dev.	0.2139	0.0573	0.1409	0.0000	0.0156	0.0188	<b>0.4830</b>
Area of SDE ( $km^2$ )							
Avg	0	312	8	0	84	172	<b>49</b>
St. dev.	0	1150	40	0	287	557	<b>375</b>
Weighted Avg x ( $km$ )							
Avg	367	372	371	374	356	377	<b>368</b>
Sd	77	73	70	61	66	54	<b>71</b>
Weighted Avg y ( $km$ )							
Avg	5,670	5,671	5,670	5,667	5,670	5,668	<b>5,670</b>
Sd	16	18	18	16	17	15	<b>17</b>

Table A.3: Descriptive statistics of cluster results based on the proposed spatio-temporal dissimilarity measure defined in section 3.3.3 for the sequences in the short form

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	All
# of sequences	102	126	145	36	49	136	<b>594</b>
# of episodes per sequence							
Avg	4.82	4.91	4.89	4.33	4.24	4.99	<b>4.82</b>
Sd	1.79	1.84	2.28	1.60	1.67	2.24	<b>2.02</b>
Eccentricity							
Avg	0.6351	0.6655	0.6025	0.4999	0.4894	0.6824	<b>0.6242</b>
Sd	0.4816	0.4725	0.4878	0.5070	0.5047	0.4658	<b>0.4830</b>
Area of SDE ( $km^2$ )							
Avg	89	44	57	20	17	32	<b>49</b>
Sd	723	191	380	115	78	179	<b>375</b>
Weighted Avg x ( $km$ )							
Avg	287	388	346	224	491	428	<b>368</b>
Sd	18	13	17	39	24	12	<b>71</b>
Weighted Avg y ( $km$ )							
Avg	5,685	5,664	5,678	5,689	5,653	5,656	<b>5,670</b>
Sd	15	10	12	14	7	9	<b>17</b>

Table A.4: Descriptive statistics of cluster results based on the existing distance measure described in section 3.3.2 for the sequences in the short form



	<b>Proposed measure</b>	<b>Existing measure</b>
Area of SDE	0.0754**	0.2614
Eccentricity	0.0000*	0.3081
Weigthed Avg x	0.9620	0.0000*
Weighted Avg y	0.3798	0.0000*

\* Significant on 0.05 level

\*\* Significant on 0.10 level

Table A.5: P-values of ANOVA tests for the sequences in the short form

### A.1.2 Design of Socio-Demographical Profiles

These clustering results now serve as a basis for designing the classification tree visualized in figures A.3 and A.4 and table A.6. In the case of the short form sequences, the classification tree indicates that the number of working hours is the most important variable in breaking down the sequences. The sequences found in cluster id 1 are mainly linked to individuals working less than 20.5 hours per week, which roughly corresponds to working less than half time. The tree further distinguishes between clusters 1, 2 and 3, based on weekdays vs. weekend days. As the share of clusters 4, 5 and 6 within the whole dataset is too small compared to the other clusters, the tree is not able to assign a socio-demographical profile to these clusters.

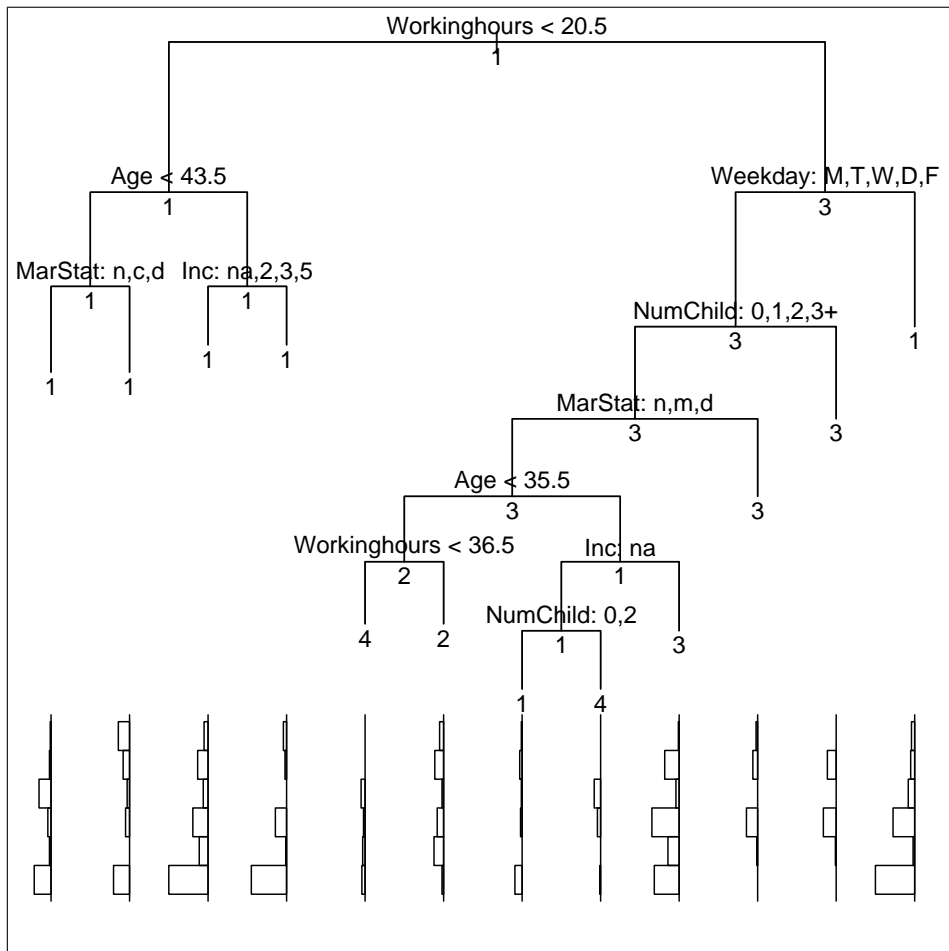


Figure A.3: Decision tree attaching socio-demographical profiles to the cluster results for the sequences in the short form (labels for weekday: *M* = Monday, *T* = Tuesday, *W* = Wednesday, *D* = Thursday, *F* = Friday, *S* = Saturday, *Z* = Sunday; labels for income: 0 = < 750, 1 = 750 – 1250, 2 = 1250 – 1750, 3 = 1750 – 2250, 4 = 2250 – 2750, 5 = > 2750; labels for marital status: *n* = notmarried, *c* = cohabit, *m* = married, *d* = divorced, *w* = widow(er))

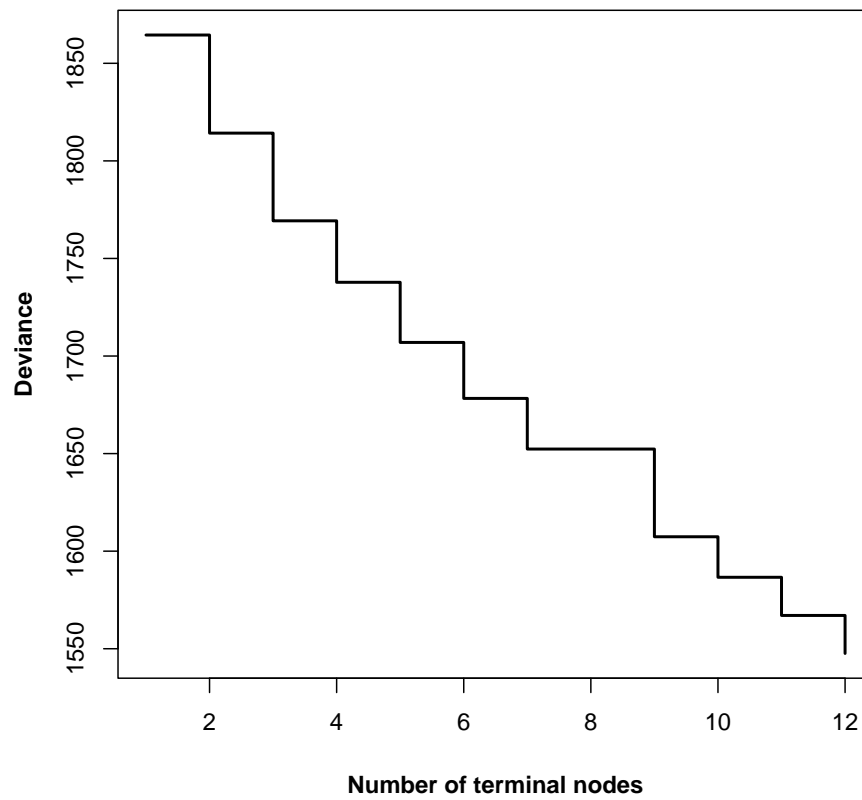


Figure A.4: Overall standard deviation of predictor variable (i.e. cluster id) for the sequences in the short form

Test rule	#cases	Pred.	%clu1	%clu2	%clu3	%clu4	%clu5	%clu6	Leaf
Root	594	1	0.3872	0.0791	0.2508	0.0875	0.1330	0.0623	
<i>Workinghours</i> < 20.5	263	1	0.2256	0.0236	0.0707	0.0404	0.0421	0.0404	
<i>Age</i> < 43.5	95	1	0.0690	0.0034	0.0151	0.0303	0.0168	0.0253	
<i>Mar.stat.</i> : <i>coh., not mar., div.</i>	45	1	0.0354	0.0034	0.0067	0.0253	0.0034	0.0017	*
<i>Mar.stat.</i> : <i>na, mar.</i>	50	1	0.0337	0.0000	0.0084	0.0051	0.0135	0.0236	*
<i>Age</i> > 43.5	167	1	0.1556	0.0201	0.0552	0.0100	0.0251	0.0151	
<i>Inc.</i> : <i>na, 1250 – 2250, &gt; 2750</i>	103	1	0.0825	0.0185	0.0320	0.0101	0.0219	0.0084	*
<i>Inc.</i> : < 1250, 2250 – 2750	65	1	0.0741	0.0017	0.0236	0.0000	0.0034	0.0067	*
<i>Workinghours</i> > 20.5	331	3	0.1616	0.0556	0.1802	0.0471	0.0909	0.0219	
<i>Weekday</i> : <i>workday</i>	236	3	0.0791	0.0522	0.1347	0.0337	0.0825	0.0151	
<i>#child.</i> : 0, 1, 2, 3+	208	3	0.0791	0.0505	0.1077	0.0337	0.0640	0.0152	
<i>Mar.stat.</i> : <i>(not)mar., div.</i>	185	3	0.0791	0.0488	0.0842	0.0337	0.0539	0.0118	
<i>Age</i> < 35.5	54	2	0.0101	0.0253	0.0168	0.0118	0.0185	0.0084	
<i>Work.hours</i> < 36.5	14	4	0.0067	0.0051	0.0034	0.0084	0.0000	0.0000	*
<i>Work.hours.</i> > 36.5	40	2	0.0034	0.0202	0.0135	0.0337	0.0185	0.0084	*
<i>Age</i> > 35.5	131	1	0.0690	0.0236	0.0673	0.0219	0.0354	0.0034	
<i>Inc.</i> : <i>na</i>	29	1	0.0168	0.0000	0.0101	0.0151	0.0050	0.0017	
<i>#child.</i> : 0, 2	16	1	0.0152	0.0000	0.0034	0.0017	0.0051	0.0017	*
<i>#child.</i> : 1	13	4	0.0017	0.0000	0.0067	0.0135	0.0000	0.0000	*
<i>Inc.</i> : <i>not na</i>	102	3	0.0522	0.0236	0.0572	0.0067	0.0303	0.0017	*
<i>Mar.stat.</i> : <i>na, coh., wid.</i>	23	3	0.0000	0.0017	0.0236	0.0000	0.0101	0.0034	*
<i>#child.</i> : 3	28	3	0.0000	0.0017	0.0269	0.0000	0.0185	0.0000	*
<i>Weekday</i> : <i>weekend</i>	95	1	0.0825	0.0034	0.0455	0.0135	0.0084	0.0067	*

Table A.6: Outcome of decision tree attaching socio-demographical profiles to the cluster results for the sequences in the short form

## Bibliography

- Algers, Staffan, Eliasson, Jonas, & Mattsson, Lars-Göran. 2005. Is It Time to Use Activity-Based Urban Transport Models? A Discussion of Planning Needs and Modelling Possibilities. *The Annals of Regional Science*, **39**(4), 767–789.
- Anderson, Rebekah, Al-Akhras, Ahmad, & Gill, Nicolas. 2003. Implementation of a Tour-Based Microsimulation Regional Travel Demand Model. *Pages 12–24 of: 9th TRB Conference on the Application of Transportation Planning Methods, Session 1: Innovations in Travel Modeling*.
- Arentze, Theo A., & Timmermans, Harry J.P. 2003. Modelling Learning and Adaptation Processes in Activity-Travel Choice. *Transportation*, **30**(1), 37–62.
- Arentze, Theo A., & Timmermans, Harry J.P. 2004. A Learning-Based Transportation Oriented Simulation System. *Transportation Research Part B: Methodological*, **38**(7), 613–633.
- Arentze, Theo A., & Timmermans, Harry J.P. 2005a. *ALBATROSS - version 2.0: a Learning-Based Transportation Oriented Simulation System*. 1st edn. EIRASS, Technische Universiteit Eindhoven, The Netherlands.
- Arentze, Theo A., & Timmermans, Harry J.P. 2005b. Modelling Learning and Adaptation in Transportation Contexts. *Transportmetrica*, **1**(Special issue: Some Recent Advances in Transportation Studies), 13–22.
- Arentze, Theo A., Pelizaro, Claudia, & Timmermans, Harry J.P. 2005. Implementation of a Model of Dynamic Activity-Travel Rescheduling Decisions: an Agent-Based Micro-

## BIBLIOGRAPHY

---

- Simulation Framework. *In: 9th International Conference on Computers in Urban Planning and Urban Management (CUPUM)*.
- Bachi, Roberto. 1963. Standard Distance Measures and Related Methods for Spatial Analysis. *Papers of the Regional Science Association*, **10**(1), 83–132.
- Bellemans, Tom, Kochan, Bruno, Janssens, Davy, & Wets, Geert. 2008. In the Field Evaluation of the Impact of a GPS-Enabled Personal Digital Assistant on Activity-Travel Diary Data Quality. *In: 87th Annual Meeting of the Transportation Research Board (TRB)*.
- Bhat, Chandra R., Guo, Jessica Y., Srinivasan, Sivaramakrishnan, & Sivakumar, Aruna. 2003. *Activity-Based Travel-Demand Modeling for Metropolitan Areas in Texas: a Micro-Simulation Framework for Forecasting*. Tech. rept. FHWA/TX-03/4080-4, Centre for Transportation Research, University of Texas.
- Bhat, Chandra R., Guo, Jessica Y., Srinivasan, Sivaramakrishnan, & Sivakumar, Aruna. 2004. A Comprehensive Econometric Micro-Simulator for Daily Activity-Travel Patterns (CEMDAP). *Transportation Research Record: Journal of the Transportation Research Board*, **1894**, 57–66.
- Borgers, Aloys W.J., Timmermans, Harry J.P., & van der Waerden, Peter J.H.J. 2002. Patricia: Predicting Activity-Travel Interdependencies with a Suit of Choice-Based, Interlinked Analyses. *Transportation Research Record: Journal of the Transportation Research Board*, **1807**, 145–153.
- Bowman, John L. 1995. *Activity Based Travel Demand Model System with Daily Activity Schedules*. Masterthesis, Massachusetts Institute of Technology.
- Bowman, John L. 1998. *The Day Activity Schedule Approach to Travel Demand Analysis*. Doctoral dissertation, Massachusetts Institute of Technology.
- Breiman, Leo, Friedman, Jerome, Olshen, Richard A., & Stone, Charles J. 1984. *Classification and Regression Trees*. 1st edn. Belmont, California: Wadsworth.

- Buliung, Ron N., & Rimmel, Tarmo K. 2008. Open Source, Spatial Analysis, and Activity-Travel Behaviour Research: Capabilities of the ASPACE Package. *Journal of Geographical Systems*, **10**(2), 191–216.
- Chapin, F. Stuart. 1974. *Human Activity Patterns in the City*. 1st edn. New York: John Wiley and Sons.
- Charypar, David, & Nagel, Kai. 2005. Q-Learning for Flexible Learning of Daily Activity Plans. *In: 84th Annual Meeting of the Transportation Research Board (TRB)*.
- Charypar, David, Graf, Philip, & Nagel, Kai. 2004. Q-Learning for Flexible Learning of Daily Activity Plans. *In: 4th Swiss Transport Research Conference (STRC)*.
- Doherty, Sean T. 2006. Should We Abandon Activity Type Analysis? Redefining Activities by Their Salient Attributes. *Transportation*, **33**(5), 517–536.
- Doherty, Sean T., Miller, Eric J., Axhausen, Kay W., & Gärling, Tommy. 2002. A Conceptual Model of the Weekly Household Activity-Travel Scheduling Process. *Pages 149–165 of: Stern, E., Salomon, I., & Bovy, P. (eds), Travel Behaviour: Patterns, Implications and Modelling*, 1st edn. Cheltenham, UK: Elgar Publishing Ltd.
- Driessens, Kurt. 2004. *Relational Reinforcement Learning*. Doctoral dissertation, Katholieke Universiteit Leuven.
- Ettema, Dick, & Timmermans, Harry J.P. 1997a. *Activity-Based Approaches to Travel Analysis*. 1st edn. Oxford, UK: Elsevier Science Ltd.
- Ettema, Dick, Borgers, Aloys W.J., & Timmermans, Harry J.P. 1996. Simulation Model of Activity Scheduling Behavior. *Transportation Research Record: Journal of the Transportation Research Board*, **1551**, 1–11.
- Ettema, Dick F., & Timmermans, Harry J.P. 1997b. Theories and Models of Activity Patterns. *Pages 1–36 of: Ettema, Dick, & Timmermans, Harry J.P. (eds), Activity-Based Approaches to Travel-Analysis*, 1st edn. Oxford: Pergamon.

## BIBLIOGRAPHY

---

- Fried, Marc A., Havens, John, & Thall, Matthew. 1977. *Travel Behavior - A Synthesized Theory*. Tech. rept. National Cooperative Highway Research Program - Transportation Research Board - National Research Council.
- Fujii, Satoshi, Kitamura, Ryuichi, & Monma, Toshiyuki. 1998. A Utility-Based Micro-Simulation Model System of Individuals' Activity-Travel Patterns. *In: 77th Annual Meeting of the Transportation Research Board (TRB)*.
- Gärbling, Tommy, Gillholm, Robert, Romanus, Joakim, & Selart, Marcus. 1997. Interdependent Activity and Travel Choices: Behavioural Principles of Integration of Choice Outcomes. *Pages 135-150 of: Ettema, Dick, & Timmermans, Harry J.P. (eds), Activity-Based Approaches to Travel-Analysis*, 1st edn. Oxford: Pergamon.
- Goodwin, Phil, Kitamura, Ryuichi, & Meurs, Henk. 1990. Some Principles of Dynamic Analysis of Travel Behaviour. *Pages 56-72 of: Jones, Peter (ed), Developments in Dynamic and Activity-Based Approaches to Travel Analysis*. Aldershot, England: Gower Publishing Company.
- Hägerstrand, Torsten. 1970. What About People in Regional Science? *Papers of the Regional Science Association*, **24**(1), 7-24.
- Hoppner, Frank, Klawonn, Frank, Kruse, Rudolf, & Runkler, Thomas. 1999. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Chichester: John Wiley and Sons Ltd.
- Janssens, Davy. 2005. *Calibrating Unsupervised Machine Learning Algorithms for the Prediction of Activity-Travel Patterns*. Doctoral dissertation, Hasselt University.
- Janssens, Davy, & Wets, Geert. 2005. The Presentation of an Activity-Based Approach for Surveying and Modelling Travel Behaviour. *Pages 1935-1945 of: 32nd Colloquium Vervoersplanologisch Speurwerk 2005: Duurzame mobiliteit: Hot or not?*, vol. 32(2).
- Janssens, Davy, Lan, Yu, Wets, Geert, & Chen, Guoqing. 2005. Optimizing Activity-Travel Sequences by Means of Reinforcement Learning. *In: BIVÉC-GIBET Transport Research Day*.



- Joh, Chang-Hyeon. 2004. *Measuring and Predicting Adaptation in Multidimensional Activity-Travel Patterns*. Doctoral dissertation, Eindhoven University.
- Joh, Chang-Hyeon, Arentze, Theo A., & Timmermans, Harry J.P. 2001. Pattern Recognition in Complex Activity-Travel Patterns: A Comparison of Euclidean Distance, Signal Processing Theoretical, and Multidimensional Sequence Alignment Methods. *In: 80th Annual Meeting of the Transportation Research Board (TRB)*.
- Joh, Chang-Hyeon, Arentze, Theo a., Hofman, Frank, & Timmermans, Harry J.P. 2002. Activity Pattern Similarity: a Multidimensional Sequence Alignment Method. *Transportation Research Part B: Methodological*, **36**(2), 385–403.
- Joh, Chang-Hyeon, Arentze, Theo A., & Timmermans, Harry J.P. 2004. Activity-Travel Scheduling and Rescheduling Decision Processes: Empirical Estimation of Aurora Model. *Transportation Research Record: Journal of the Transportation Research Board*, **1898**, 10–18.
- Joh, Chang-Hyeon, Arentze, Theo A., & Timmermans, Harry J.P. 2007. Identifying Skeletal Information of Activity Patterns of a Group. *In: 86th Annual Meeting of the Transportation Research Board (TRB)*.
- Jones, Peter M. 1979. New Approaches to Understanding Travel Behaviour: The Human Activity Approach. *Pages 55–80 of: Hensher, D.A., & Stopher, P.R. (eds), Behavioural Travel Modelling*, 1st edn. London: Groom Helm Ltd.
- Jones, Peter M., Dix, Martin C., Clarke, Mike I., & Heggie, Ian G. 1983. *Understanding Travel Behaviour*. 1st edn. Aldershot, England: Gower Publishing Company Limited.
- Kaelbling, Leslie P., Littman, Michael L., & Moore, Andrew W. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.
- Kaufman, Leonard, & Rousseeuw, Peter J. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, New Jersey, USA: John Wiley & Sons Inc.

## BIBLIOGRAPHY

---

- Kitamura, Ryuichi. 1996. Applications of Models of Activity Behavior for Activity Based Demand Forecasting. *In: Activity-Based Travel Forecasting Conference: Summary, Recommendations and Compendium of Papers, Texas Transportation Institute.*
- Kitamura, Ryuichi, & Fujii, Satoshi. 1998. Two Computational Process Models of Activity-Travel Behavior. *Pages 251–279 of: Gärling, Tommy, Laitila, Thomas, & Westin, Kerstin (eds), Theoretical Foundations of Travel Choice Modeling, 1st edn. Elsevier.*
- Kolter, Jeremy Z., & Maloof, Marcus A. 2005. Using Additive Expert Ensembles to Cope with Concept Drift. *Pages 449–456 of: 22nd International Conference on Machine Learning.* ACM International Conference Proceeding Series, vol. 119. Bonn, Germany: ACM, New York, NY, USA.
- Kulkarni, Anup, & McNally, Michael G. 2001. A Microsimulation of Daily Activity Patterns. *In: 80th Annual Meeting of the Transportation Research Board (TRB).*
- Kwan, Mei-Po. 1997. Gisicas: An Activity-Based Travel Decision Support System Using a GIS-Interfaced Computational-Process Model. *Pages 263–282 of: Ettema, D., & Timmermans, Harry (eds), Activity-Based Approaches to Activity Analysis, 1st edn. Pergamon.*
- Kwan, Mei-Po. 2000. Interactive Geovisualization of Activity-Travel Patterns Using Three-Dimensional Geographical Information Systems: a Methodological Exploration with a Large Data Set. *Transportation Research Part C: Emerging Technologies, 8(1–6), 185–203.*
- Lefever, D. Welty. 1926. Measuring Geographic Concentration by Means of the Standard Deviation Ellipse. *American Journal of Sociology, 32(1), 88–94.*
- Littman, Michael, & Boyan, Justin A. 1993. A Distributed Reinforcement Learning Scheme for Network Routing. *Pages 45–51 of: Alspector, J., Goodman, R., & Brown, T.X. (eds), International Workshop on Applications of Neural Networks to Telecommunications.* Princeton, USA: Lawrence Erlbaum Associates, Hillsdale, NJ, USA.
- Mataric, Maja J. 1994. Reward Functions for Accelerated Learning. *Pages 181–189 of:*

- Cohen, William W., & Hirsh, Haym (eds), *Eleventh International Conference on Machine Learning*. New Brunswick, NJ, USA: Morgan Kaufmann Publishers.
- McIntosh, John, & Yuan, May. 2005. Assessing Similarity of Geographic Processes and Events. *Transactions in GIS*, **9**(2), 223-245.
- McNally, Michael G. 2000 (Dec.). *The Activity-Based Approach*. Tech. rept. UCI-ITS-AS-WP-00-4. Center for Activity Systems Analysis., Irvine, California.
- McNally, Michael G. 2008 (Nov.). *The Four Step Model*. Tech. rept. UCI-ITS-AS-WP-07-2. Center for Activity Systems Analysis, Irvine.
- Mitchell, T.M. 1997a. *Machine Learning*. USA: The McGrawhill Companies, Inc.
- Mitchell, Tom M. 1997b. Reinforcement Learning. *Pages 367-392 of: Machine Learning*. USA: The McGrawhill Companies, Inc.
- Pas, Eric I. 1983. A Flexible and Integrated Methodology for Analytical Classification of Daily Activity-Travel Behavior. *Transportation Science*, **17**(4), 405-429.
- Pendyala, Ram M., & Bhat, Chandra R. 2006. Validation and Assessment of Activity-Based Travel Demand Modeling Systems. *In: Innovations in Travel Modelling Conference*.
- Potts, Duncan, & Sammut, Claude. 2005. Incremental learning of linear model trees. *Machine Learning*, **61**(1-3), 5-48.
- Quinlan, John R. 1993. *C4.5: programs for machine learning*. 1st edn. San Mateo, California: Kaufmann.
- Recker, Will W., McNally, Michael G., & Root, Gregory S. 1983. A Methodology for Activity-Based Travel Analysis: The STARCHILD Model. *In: 13th North American Meeting of the Regional Science Association*.
- Schlich, R. 2001. Measurement Issues in Identifying Variability in Travel Behaviour. *In: 1st Swiss Transport Research Conference*.

## BIBLIOGRAPHY

---

- Schlich, Robert, & Axhausen, Kay W. 2004. *Analysing Interpersonal Variability for Homogeneous Groups of Travellers*. Tech. rept. Arbeitsbericht Verkehrs- und Raumplanung, 296, IVT, ETH.
- Shiftan, Yoram, & Surhbier, John. 2002. The Analysis of Travel and Emission Impacts of Travel Demand Management Strategies Using Activity-Based Models. *Transportation*, **29**(2), 145–168.
- Shiftan, Yoram, Ben-Akiva, Moshe E., Prousaloglou, Kimon, de Jong, Gerard, Popuri, Yasasvi, Kasturirangan, Krishnan, & Bekhor, Shlomo. 2003. Activity-Based Modeling as a Tool for Better Understanding Travel Behaviour. *In: 10th International Conference on Travel Behaviour Research (IATBR)*.
- Smart, William D., & Kaelbling, Leslie P. 2000. Practical Reinforcement Learning in Continuous Spaces. *Pages 903–910 of: 17th International Conference on Machine Learning (ICML'00)*.
- Stead, Dominic, & Banister, David. 2001. Influencing Mobility Outside Transport Policy. *Innovation: The European Journal of Social Sciences*, **14**(4), 315–330.
- Sutton, Richard S., & Barto, Andrew G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts, USA/London, England: The MIT Press.
- Timmermans, Harry J.P. 2000. *ALBATROSS: a Learning-Based Transportation Oriented Simulation System*. 1st edn. Den Haag, Netherlands: Koninklijke bibliotheek.
- Timmermans, Harry J.P. 2001. Models of Activity Scheduling Behaviour. *Mobilitat und Stadt, Stadt Region Land*, 33–47.
- Timmermans, Harry J.P. 2006. Analyses and Models of Household Decision Making Processes. *In: International Association for Travel Behaviour Research (IATBR) Conference*.
- Uther, William T.B., & Veloso, Manuela M. 1998. Tree Based Discretization for Continuous State Space Reinforcement Learning. *Pages 769–774 of: AAAI '98/IAAI '98: 15th National and 10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. American Association for Artificial Intelligence.

- Vlachos, Michail, Gunopulos, D., & Das, Gautam. 2004. Rotation Invariant Distance Measure for Trajectories. *In: 10th International Conference on Knowledge Discovery and Data Mining*.
- Wang, Donggen, & Timmermans, Harry J.P. 2000. Conjoint-Based Model of Activity Engagement, Timing, Scheduling, and Stop Pattern Formation. *Transportation Research Record: Journal of the Transportation Research Board*, **1718**, 10–17.
- Watkins, Christopher J.C.H. 1989. *Learning from Delayed Rewards*. Doctoral dissertation, University of Cambridge.
- Watkins, Christopher J.C.H., & Dayan, Peter. 1992. Technical Note: Q-Learning. *Machine Learning*, **8**(3–4), 279–292.
- Widmer, Gerhard, & Kubat, Miroslav. 1996. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, **23**(1), 69–101.
- Wilson, Clarke. 1998a. Activity Pattern Analysis by Means of Sequence-Alignment Methods. *Environment and Planning A*, **30**(6), 1017–1038.
- Wilson, Clarke. 1998b. Analysis of Travel Behavior Using Sequence Alignment Methods. *Transportation Research Record: Journal of the Transportation Research Board*, **1645**, 52–59.
- Wilson, Clarke. 2001. Activity Patterns of Canadian Women: Application of ClustalG Sequence Alignment Software. *Transportation Research Record: Journal of the Transportation Research Board*, **1777**, 55–67.
- Wilson, Clarke. 2008. Activity Patterns in Space and Time: Calculating Representative Hagerstrand Trajectories. *Transportation*, **35**(4), 485–499.
- Witten, Ian H., & Frank, Eibe. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, USA: Morgan Kaufmann Publishers.
- Yuill, Robert S. 1971. The Standard Deviation Ellipse: an Updated Tool for Spatial Description. *Geographiska Annaler Series B Human Geography*, **53**(1), 28–39.

## BIBLIOGRAPHY

---

- Zahavi, Yacov, & Ryan, James M. 1980. The Stability of Travel Components over Time. *Transportation Research Record: Journal of the Transportation Research Board*, **200**, 19–26.
- Zahavi, Yacov, & Talvitie, Antti. 1980. Regularities in Travel Time and Money Expenditures. *Transportation Research Record: Journal of the Transportation Research Board*, **750**, 13–19.
- Zennir, Youcef, & Couturier, Pierre. 2005. Multiactor approach and hexapod robot learning. *In: IEEE International Symposium on Computational Intelligence in Robotics and Automations*.







# CURRICULUM VITAE

## PERSONAL INFORMATION

---

VANHULSEL, Marlies  
Keizel 9  
B-3590 Diepenbeek, Belgium  
+32 494 575 369  
marliesvanhulsel@gmail.com

## COMPETENCES

---

Analytical  
Result-oriented  
Inquisitive  
Fast learner  
Autonomous  
Structured  
Strong affinity with numbers

## EDUCATION

---

*2005 - 2009 Hasselt University Diepenbeek*  
PhD in Traffic Science  
*2004-2006 PCVO College of Education Diepenbeek*  
Certificate of Pedagogical Competences  
*2003 - 2004 Hasselt University Diepenbeek*  
Public Law - Relations with the Government  
*1999 - 2004 Hasselt University Diepenbeek*  
Commercial Engineering and Management Informatics

## **WORK EXPERIENCE**

---

*2009 - now Vito Mol*

Applied research sector

Researcher

Data analysis and modelling

Modelling emission

*2005 - 2009 Hasselt University Diepenbeek*

Research and education sector

Researcher/PhD Candidate

Data mining and data analysis

Modelling activity-travel sequences

*2004-2005 Luminus Hasselt*

Energy sector

Pricing analyst

Market research

Price analyses and price fixing

## **COMPUTER SKILLS**

---

Microsoft Office: proficient

Statistical packages (R, SPlus): independent

Statistical packages (SAS, SPSS): basic

Programming languages (HTML, PHP, VisualC++): independent

## **LANGUAGES**

---

Dutch: mother tongue

French: independent

English: proficient

Spanish: basic

## PUBLICATIONS

---

Beckx, C., L. Int Panis, **M. Vanhulsel**, G. Wets, and R. Torfs (2007). Gender-Linked Disparity in Vehicle Exhaust Emissions? Results from an Activity-Based Survey. In G. M. Morrison and S. Rauch (Eds.), *8th Highway and Urban Environment Symposium*, Volume 12 of *Highway and Urban Environment*, Nicosia, Cyprus, pp. 594. Springer.

Hannes, E., F. Liu, **M. Vanhulsel**, D. Janssens, T. Bellmans, K. Vanhoof and G. Wets (2009). Tracking Household Routines Using Scheduling Hypotheses Embedded in Skeletons (THRUSHES). Submitted to *Transportmetrica*.

**Vanhulsel, M.**, C. Beckx, D. Janssens, K. Vanhoof, and G. Wets (2009). Measuring Dissimilarity of Geographically Dispersed Space-Time Paths. Submitted to *Transportation*.

**Vanhulsel, M.**, D. Janssens, K. Vanhoof, and G. Wets (2008). Application of On-line Regression Tree Induction to Forecast Traffic Flows. In *First Ubiquitous Knowledge Discovery Workshop (UKD08) (Part of ECML/PKDD 2008 Conferences)*, Antwerp, Belgium.

**Vanhulsel, M.**, D. Janssens, and G. Wets (2006). Generating dynamic activity-travel schedules. In *PlanSIG 2006, 25th Workshop of the UK Planning and Scheduling Special Interest Group*, Nottingham, U.K.

**Vanhulsel, M.**, D. Janssens, and G. Wets (2007). Calibrating a New Reinforcement Learning Mechanism for Modeling Dynamic Activity-Travel Behavior and Key Events. In *86th Annual Meeting of the Transportation Research Board (TRB)*, Washington D.C., USA.

**Vanhulsel, M.**, D. Janssens, and G. Wets (2007). Use of a relational reinforcement learning algorithm to generate dynamic activity-travel patterns. In *TRISTAN, 6th Triennial Symposium on Transportation Analysis*, Phuket Island, Thailand.

**Vanhusel, M.**, D. Janssens, G. Wets, and K. Vanhoof (2008). Implementing an Improved Reinforcement Learning Algorithm for the Simulation of Weekly Activity-Travel Sequences. In *87th Annual Meeting of the Transportation Research Board (TRB)*, Washington D.C., USA.

**Vanhusel, M.**, D. Janssens, G. Wets, and K. Vanhoof (2008). Simulation of Sequential Data: An Enhanced Reinforcement Learning Approach. *Expert Systems With Applications* 36 (4), 8032-8039.



