Doctoral Thesis

Choosing the daily schedule
Expanding activity based travel demand modelling

Author(s):
Feil, Matthias

Publication Date:
2010

Permanent Link:
https://doi.org/10.3929/ethz-a-006200573

Rights / License:
In Copyright - Non-Commercial Use Permitted
CHOOSING THE DAILY SCHEDULE:
EXPANDING ACTIVITY-BASED TRAVEL DEMAND MODELLING

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

MATTHIAS FEIL
Dipl. Wirt.-Ing., Universität Karlsruhe (TH)

born 27 February 1980

citizen of Germany

accepted on the recommendation of

Prof. Dr. Kay W. Axhausen
Prof. Dr. Yoram Shiftan

2010
Contents

Abstract xii

Zusammenfassung xiii

Acknowledgements xiv

1 Introduction 1

1.1 Need for modelling transport 1

1.2 Motivation for activity-based travel demand modelling 1

1.3 Objectives and innovations of the thesis 2

1.4 Methodology 3

1.5 Relevance for science and economy 4

1.6 Structure of the thesis 5

2 Theoretic background and literature review 6

2.1 Classic 4-step transport model 6

2.1.1 Overview of the four steps 7

2.1.2 Disaggregate 4-step transport models 10

2.1.3 Shortcomings of the classic 4-step transport model 10

2.2 Activity-based travel demand modelling 12

2.2.1 Core principles and definitions 12
2.2.2 Advantages .................................................. 13
2.2.3 Fundamental problem of activity scheduling ....................... 13
2.2.4 Existing modelling concepts .................................... 14
2.2.5 Integration of activity-based travel demand models and assignment models .................................................. 27
2.2.6 Discussion .................................................. 29

2.3 Introduction to MATSim ......................................... 30
2.3.1 MATSim structure and constraints ................................ 31
2.3.2 Discussion .................................................. 38

3 Comprehensive schedule optimization: PlanomatX ................. 39
3.1 Problem formulation ............................................... 39
3.2 Heuristic solution algorithm ....................................... 40
3.2.1 Initialization of the algorithm (step A) ......................... 42
3.2.2 Neighbourhood creation (step B) ............................. 43
3.2.3 Tabu check (step C) ........................................ 52
3.2.4 Optimization of neighbourhood solutions (step D) .......... 53
3.2.5 Selection of best neighbourhood schedule solution (step E) .. 58
3.2.6 Finish of the algorithm (step F) ............................... 58
3.3 Test results of PlanomatX with Planomat ......................... 59
3.3.1 Fictive chessboard scenario .................................. 59
3.3.2 Test I: Test of PlanomatX on chessboard scenario .......... 61
3.4 Discussion .................................................. 67

4 Optimization of mode choices and activity timings: TimeModeChoicer ..... 68
4.1 Problem formulation ............................................... 68
4.2 Heuristic solution algorithm - Optimization of activity timings .. 69
4.2.1 Initialization of the algorithm (step A) ............................................. 69
4.2.2 Neighbourhood creation (step B) ...................................................... 70
4.2.3 Selection of best neighbourhood schedule solution (step C) .............. 74
4.2.4 Finish of the algorithm (step D) ....................................................... 75

4.3 Heuristic solution algorithm - Integration of mode choice ..................... 75
4.3.1 Impact of mode choice dimension on solution space ......................... 75
4.3.2 TimeModeChoicer’s four alternatives to integrate mode choice .......... 76
4.3.3 A-priori assumptions for walking ................................................... 78

4.4 Aligned interface between TimeModeChoicer and PlanomatX ............... 79
4.4.1 Feasibility of schedules ................................................................. 79
4.4.2 Employment of router strategy ...................................................... 79
4.4.3 Employment of scoring module ..................................................... 80

4.5 Test results of the TimeModeChoicer as a stand-alone replanning strategy ................................................. 80
4.5.1 Test II: Test of TimeModeChoicer on chessboard scenario .............. 80
4.5.2 Test III: Variation of TimeModeChoicer settings on chessboard scenario 85

4.6 Test results of PlanomatX with TimeModeChoicer ............................. 88
4.6.1 Test IV: Test of PlanomatX with TimeModeChoicer on chessboard sce-

nario ........................................................................................................ 88
4.6.2 Test V: Variation of PlanomatX/TimeModeChoicer settings on chess-

board scenario ....................................................................................... 92
4.6.3 Greater Zurich scenario ................................................................. 96
4.6.4 Test VI: Test of PlanomatX on greater Zurich scenario .................. 96

4.7 Discussion ......................................................................................... 104

5 Schedule recycling ............................................................................. 105
5.1 Problem formulation ........................................................................ 105
5.2 Solution algorithm ........................................................................... 105
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Classic 4-step model structure</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Space-time diagram and prisms</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Characteristics of solution search and choice</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Basic model structure of activity-based transport models</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>Integration of activity-based travel demand and assignment models</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>MATSim architecture</td>
<td>31</td>
</tr>
<tr>
<td>3.1</td>
<td>PlanomatX as a new MATSim replanning strategy</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>PlanomatX process flow</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Input data to PlanomatX algorithm</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>PlanomatX process flow with focus on neighbourhood moves</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>Change-number sub-algorithm</td>
<td>45</td>
</tr>
<tr>
<td>3.6</td>
<td>Activity insertion operator</td>
<td>46</td>
</tr>
<tr>
<td>3.7</td>
<td>Example of invalid insertion</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>Change-order sub-algorithm</td>
<td>49</td>
</tr>
<tr>
<td>3.9</td>
<td>Activity swap operator</td>
<td>50</td>
</tr>
<tr>
<td>3.10</td>
<td>Change-type sub-algorithm</td>
<td>51</td>
</tr>
<tr>
<td>3.11</td>
<td>Change-type operator</td>
<td>51</td>
</tr>
<tr>
<td>3.12</td>
<td>PlanomatX's inner loop</td>
<td>53</td>
</tr>
<tr>
<td>3.13</td>
<td>Reduced location choice</td>
<td>55</td>
</tr>
</tbody>
</table>
3.14 Chessboard scenario ........................................... 60
3.15 Example of knowledge and initial schedule in chessboard scenario .... 60
3.16 Plot of utility functions for all tests on the chessboard scenario ......... 62
3.17 Test I: Development of average executed schedule utility, trip distance and trip travel time ....................................................... 64
3.18 Test I: Initial demand schedule and optimized schedules of test agent . 65
3.19 Test I: Runtime shares per MATSim iteration .......................... 66
4.1 TimeModeChoicer process flow - optimization of activity timings ........ 70
4.2 TimeModeChoicer's neighbourhood process steps ......................... 71
4.3 TimeModeChoicer's neighbourhood creation ................................ 72
4.4 Swap operator .................................................................. 74
4.5 Mode choice for affected sub-tour(s) only .................................. 77
4.6 Alternatives of mode choice in TimeModeChoicer ......................... 78
4.7 Test II: Development of average executed schedule utility, trip distance and trip travel time ....................................................... 82
4.8 Test II: Comparison of average schedules ................................... 83
4.9 Test II: Initial demand schedule and optimized test agent schedules .... 84
4.10 Test II: Runtime shares per MATSim iteration .......................... 84
4.11 Test III: Impact of variation of TimeModeChoicer's neighbourhood size on development of utility scores and runtimes ................. 85
4.12 Test III: Impact of variation of TimeModeChoicer's number of iterations and stop criterion on development of utility scores and runtimes .... 86
4.13 Test III: Impact of variation of TimeModeChoicer's offset time on development of utility scores and runtimes .......................... 87
4.14 Test III: Impact of variation of mode choice on development of utility scores and runtimes ...................................................... 87
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.15</td>
<td>Test IV: Development of average executed schedule utility, trip distance and trip travel time</td>
<td>89</td>
</tr>
<tr>
<td>4.16</td>
<td>Test IV: Initial demand schedule and optimized schedules of test agent</td>
<td>90</td>
</tr>
<tr>
<td>4.17</td>
<td>Test IV: Runtime shares per MATSim iteration</td>
<td>91</td>
</tr>
<tr>
<td>4.18</td>
<td>Test V: Impact of PlanomatX’s neighbourhood size variation on development of utility score and runtime</td>
<td>92</td>
</tr>
<tr>
<td>4.19</td>
<td>Test V: Impact of variation of PlanomatX’s number of iterations on development of utility score and runtime</td>
<td>93</td>
</tr>
<tr>
<td>4.20</td>
<td>Test V: Impact of variation of location choice on utility score development and runtime</td>
<td>94</td>
</tr>
<tr>
<td>4.21</td>
<td>Test V: Impact of joint application of runtime-increasing PlanomatX and Time-ModeChoicer settings</td>
<td>95</td>
</tr>
<tr>
<td>4.22</td>
<td>Test V: Impact of joint application of runtime-decreasing PlanomatX and Time-ModeChoicer settings</td>
<td>96</td>
</tr>
<tr>
<td>4.23</td>
<td>Greater Zurich scenario: 30km circle around Zurich city centre</td>
<td>97</td>
</tr>
<tr>
<td>4.24</td>
<td>Test VI: Development of average executed schedule utility, trip distance and trip travel time</td>
<td>98</td>
</tr>
<tr>
<td>4.25</td>
<td>Test VI: Reported and simulated activity and travel statistics</td>
<td>99</td>
</tr>
<tr>
<td>4.26</td>
<td>Test VI: Reported and simulated number of trips and mode shares</td>
<td>101</td>
</tr>
<tr>
<td>4.27</td>
<td>Test VI: Observed and simulated traffic counts</td>
<td>102</td>
</tr>
<tr>
<td>4.28</td>
<td>Test VI: Runtime shares per MATSim iteration</td>
<td>103</td>
</tr>
<tr>
<td>5.1</td>
<td>Schedule recycling process flow</td>
<td>106</td>
</tr>
<tr>
<td>5.2</td>
<td>Process flow of assignment module</td>
<td>107</td>
</tr>
<tr>
<td>5.3</td>
<td>Distance metric process flow</td>
<td>111</td>
</tr>
<tr>
<td>5.4</td>
<td>Test VII: Development of average executed schedule utility, trip distance and trip travel time</td>
<td>114</td>
</tr>
<tr>
<td>5.5</td>
<td>Test VII: Initial demand schedule and optimized schedules of test agent</td>
<td>116</td>
</tr>
<tr>
<td>5.6</td>
<td>Test VII: Runtime shares per MATSim iteration</td>
<td>117</td>
</tr>
</tbody>
</table>
5.7 Test VIII: Development of average executed schedule utility, trip distance and trip travel time .......................................................... 119
5.8 Test VIII: Reported and simulated activity and travel statistics .......... 121
5.9 Test VIII: Reported and simulated number of trips and mode shares .... 122
5.10 Test VIII: Observed and simulated traffic counts .......................... 123
5.11 Test VIII: Runtime shares per MATSim iteration .......................... 125
5.12 Test VIII: Breakdown of runtime consumption of a schedule recycling replanning period ......................................................... 125

6.1 Dilemma of MATSim’s existing utility function ............................. 128
6.2 Generic illustration of new utility function for activity performance .... 130
6.3 Estimation methodology of utility function parameters .................. 131
6.4 Comparison of average trip distances and travel times .................. 134
6.5 Plot of the estimated utility function ........................................... 138
6.6 Analysis of home activity type ..................................................... 140
6.7 Comparison of linear dissimilarity term and box cox transformation ... 146
6.8 Reported and simulated activity and travel statistics for estimated utility function 148
6.9 Observed and simulated traffic counts for estimated utility function .... 149
6.10 Plot of the calibrated utility function .......................................... 151
6.11 Algorithm to calculate schedules’ similarities .............................. 155
6.12 Illustration of Sequence Alignment Method .................................. 156
6.13 Comparison of a two-dimensional sequence .................................. 157
6.14 Optimal trajectories according to reimplementation of MDSAM ........ 158
6.15 Generating agents’ incomes: summary of average incomes per step .... 161
6.16 Comparison of probabilities of season ticket ownership ................. 162

7.1 Utility score and runtime performances of different tests ................. 164
List of Tables

2.1 Classification of assignment models ........................................ 9

2.2 Combinatorial problem of activity-based travel demand modelling 14

2.3 Classification of existing activity-based models .......................... 16

3.1 Parameter settings of utility function for all tests on the chessboard scenario 61

3.2 Test I: Analysis of activity chains ............................................. 65

4.1 Test IV: Analysis of activity chains ........................................... 90

4.2 Test VI: Analysis of activity chains ........................................... 100

5.1 Test VII: Development of distance metric coefficients .................. 115

5.2 Test VII: Analysis of activity chains ......................................... 116

5.3 Test VIII: Development of distance metric coefficients ................. 118

5.4 Test VIII: Analysis of activity chains ......................................... 120

6.1 Overall estimation results ....................................................... 137

6.2 Comparison of estimations with and without inner home activity type, with flexible $\beta_j$, and with flexible $\beta_j$ and $\gamma_j$ 141

6.3 Comparison of estimations with aggregate and disaggregate gender parameters 143

6.4 Summary of further socio-demographic parameters estimation results 144

6.5 Comparison of estimations with four different cost formulations ....... 145

6.6 Estimated and manually calibrated utility function parameters .......... 152
6.7 Average income by educational qualification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Abstract

Activity-based travel demand models derive travel demand from people's desire to pursue activities in time and space. They generate activity-travel schedules for individual travellers or homogeneous groups of travellers. Comprehensive activity-travel schedules hold information on which activities are performed, in which order, where, for how long, and which travel modes are used between the activities including corresponding routes.

This Ph.D. thesis was part of the agent-based microsimulation project MATSim (Multi-Agent Transport Simulation), run jointly by TU Berlin and ETH Zurich. The thesis presents three innovations:

- PlanomatX is a new scheduling algorithm based on Tabu Search that generates comprehensively optimized all-day schedules. Test results on the greater Zurich scenario with more than 170,000 agents show that PlanomatX achieves significantly better optimization results than MATSim’s existing scheduling algorithms. However, it also leads to disproportional simulation runtimes.

- Schedule recycling is a new concept that significantly reduces simulation runtimes by re-using schedules of optimized travellers for other non-optimized travellers. It relieves PlanomatX’s runtime drawback and allows generation of comprehensively optimized all-day schedules for large-scale scenarios at affordable runtimes.

- MATSim’s utility function has been adapted to cope with the enhanced functionality of PlanomatX and schedule recycling. The new utility function for performance of activities follows an S-shaped formulation. The new utility function for performance of travel is linear. Both functions consider a number of socio-economic attributes. The function parameters have been empirically estimated using an enhanced Multinomial Logit (MNL) model and manually calibrated to match reported and observed traffic data.
Zusammenfassung


Diese Doktorarbeit ist im Rahmen des Forschungsprojekts MATSim (Multi-Agent Transport Simulation) entstanden, das gemeinsam von TU Berlin und ETH Zürich betrieben wird. Sie stellt drei Innovationen vor:

- PlanomatX ist ein auf der Heuristik Tabu Search basierender Algorithmus, der vollständig optimierte Tagespläne generiert. Ergebnisse aus einem Szenario, das den Großraum Zürich beschreibt und mehr als 170.000 Agenten enthält, zeigen, dass PlanomatX erheblich bessere Optimierungsresultate erreicht als MATSim’s bisherige Algorithmen. Die Ergebnisse verdeutlichen aber auch, dass PlanomatX zu sehr langen Simulationslaufzeiten führt.


Acknowledgements

I would like to express my deep gratitude to

- my supervisor Kay W. Axhausen, Professor at the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology (ETH). He provided fundamental input and ideas, challenged my results, and always kept me on track. I am very thankful for his excellent support and supervision as well as for many productive discussions.

- my co-examiner Yoram Shiftan, Professor at the Faculty of Civil and Environmental Engineering, Technion, Israel Institute of Technology. I am very grateful for his encouragement, support, critical revision and valuable suggestions.

- Michael Balmer, Ph.D., project leader of MATSim at the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology (ETH). His patience with my questions, as well as his countless methodical hints, made life much easier. He helped solve many problems, and I am very thankful for the enjoyable team work.

- Konrad Meister and Andreas Horni, doctoral students at the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology (ETH). They were in charge of MATSim’s Planomat and location choice strategies and provided very valuable advice on how to best connect my code with these strategies.

- all other doctoral students from the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology (ETH). They provided valuable insights, helped with many software issues, and fostered a very pleasant working environment.

- the secretarial staff at the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology (ETH). They took care of all administrative questions in an extremely reliable, attentive and amiable way.

- my editor, Karen Ettlin, for her patience and hard work.

- my wife Stephanie. Her love and continuous support let me get up looking forward to the day.
Chapter 1

Introduction

1.1 Need for modelling transport

Developments in transport infrastructure, policy, and systems must ensure that transport “[...] effectively and efficiently moves people and goods, shapes urban form, affects economic vitality and impacts quality of life to meet wider social objectives” (Shiftan et al., 2007). Thus, the transport planner must deal with a complex set of goals and constraints. Modelling is a meaningful tool to support the planner in his analysis:

“A model can be defined as a simplified representation of a part of the real world - the system of interest - which concentrates on certain elements considered important for its analysis from a particular point of view” (Ortúzar and Willumsen, 2001).

A model allows examination, appraisal and forecasting of real world reactions without actually implementing the situation causing the reactions. This is particularly interesting for transport domain problems where changes in infrastructure, policy or systems are cost-intensive, long-ranging, and hard to undo if considered inappropriate or unfavourable. As the transport planner depends on well-founded analysis, modelling is an important tool in reaching accurate conclusions.

1.2 Motivation for activity-based travel demand modelling

Modelling is relevant and important but is not to be confused with planning and decision making, which can produce concrete and major change. Transport modelling is useful as an effective aid to planning and decision making, but models must directly relate to the problem:
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling 2010

sufficiently complex, yet as simple as possible, and accurately representing reality. Transport models have come a long way in fulfilling these requirements. The classic 4-step transport model dates from the 1950s and has remained more or less the same since then (Ortúzar and Willumsen 2001), computational limitations being a major reason for this. These limitations no longer exist, high-capacity computing is now available at reasonable prices.

Modelling objectives have also shifted. In the 1950s, “the expansion of transport infrastructure was of primary concern” (Kitamura 1996). Many of today’s models have to deal with travel demand management. The activity-based approach of modelling travel demand addresses the shift in model objectives while taking advantage of improved computing power. It aims for a better understanding of people’s desires for, restraints on, and possibilities to perform activities. This is because people generally do not travel just for the sake of travelling, but travel to perform some activity at the destination. Travel demand is thus a by-product of the desire or need to perform activities. Because activities take place at different locations, people need to travel between them. Understanding people’s daily activity patterns is fundamental to understanding and predicting travel demand dynamics: exactly what the activity-based approach tries to capture (see sections 2.1.3 to 2.2.2 for an in-depth motivation of activity-based transport models).

1.3 Objectives and innovations of the thesis

This thesis was conducted as part of research for the agent-based microsimulation toolkit MATSim (Multi-Agent Transport Simulation). MATSim implements an activity-based approach to travel demand modelling for large samples. An activity schedule’s utility is iteratively improved against the background of overall activity-travel costs. The costs are calculated using a suitable traffic flow simulation. The co-evolutionary learning process stops when none of the agents can further improve their schedule (see section 2.3 for more details).

Up to this point, schedules could be modified in MATSim by changing activity timings and durations, modes, locations and routes. Schedules’ activity chains (number, type and sequence of the activities) were determined at the beginning of the simulation, but kept fixed over the learning process. This constraint should be relaxed. Depending on simulation objective, agents may also be allowed to modify number, types and sequence of their activities if they can benefit from such changes. However, the release of the activity chain significantly enlarges the solution space for the learning process. Regarding MATSim’s aim to be able to handle large-scale samples, reasonable computational runtimes must be ensured. The first objective of the Ph.D. thesis has thus been to design and implement an algorithm that efficiently optimizes a schedule, including the number, type and sequence of its activities.
MATSim’s original utility function parameters, defined by Charypar and Nagel (2005), feature a log form for the performance of activities. This is problematic and results in unrealistic effects when changes in the number of activities of a schedule are allowed. When the number of activities in the schedule is a dimension of the learning process, the log form leads to a lot of very short activities due to the decreasing marginal utility of the log form. In other words, a schedule of two 30-minute activities of a certain type is always better than a schedule of one 60-minute of the same activity (see section 6). Furthermore, the parameters of the original utility function were set reasonably, but arbitrarily, and not estimated empirically. The Ph.D. project’s second objective is, therefore, to introduce a utility function that could realistically handle a flexible number of activities in a schedule and to empirically estimate the parameters of the new utility function.

Given the above purpose and objectives of the thesis, the following innovations have been achieved:

- Design and implementation of efficient scheduling algorithms, particularly
  - an algorithm (“PlanomatX”) optimizing number, type, and sequence of the activities in a schedule,
  - an algorithm (“TimeModeChoicer”) optimizing mode choice as well as timings and durations of activities in a schedule,
  - an algorithm (“Schedule Recycling”) re-using agent-specific schedules for other agents to save computational runtime.

- Specification and empirical estimation of the underlying utility function, particularly
  - disaggregation of the previous utility function attributes,
  - incorporation of additional attributes to increase the explanatory power of the utility function,
  - definition of an appropriate functional form for the utility function,
  - empirical estimation of the utility function parameters, including a calibration with traffic counts.

### 1.4 Methodology

The following methods were applied to meet the thesis objectives:

- Both PlanomatX and TimeModeChoicer are based on Tabu Search (Glover, 1989). Tabu Search is an iterative, heuristic optimization method. It enhances the performance of a
local search method by avoiding, or setting tabu, previous iterations’ chosen solutions so that cycling is impossible. Sections 3.2 and 4.2 explain Tabu Search and its application to PlanomatX and TimeModeChoicer.

- Schedule Recycling introduces three innovations: the assignment module provides non-optimized agents with best matching recycled schedules from other individually optimized agents; the reverse clustering approach defines an agent attributes distance metric underlying the assignment module; and the revolving sequence of individual agents’ schedules optimization and schedule assignment ensures minimal simulation runtimes. Section 5.2 describes all three innovations.

- Empirical estimation of the utility function parameters is based on an enhanced Multinomial Logit (MNL) model (McFadden, 1974). The model is enhanced through a dissimilarity attribute reflecting the structural dissimilarity between schedules following the path-size logit or C-logit approach (Cascetta et al., 1996). The attribute is calculated using an updated version of the Multidimensional Sequence Alignment Method (MDSAM) by Joh et al. (2002) and Joh (2004). The MNL model will be discussed in section 6.2.5, MDSAM in section 6.3.

- Some general statistical fitting methods are applied in section 6.4 to impute missing socio-economic agent attributes in the Greater Zurich simulation scenario.

### 1.5 Relevance for science and economy

Activity-based travel demand modelling is among the most active research areas in transport modelling. The work presented has taken place within the MATSim research project, enabling MATSim be part of this development. The scheduling algorithm improves MATSim’s functionality. The estimation of utility function parameters allows MATSim a more consistent relationship with empirical data.

Sections 1.1 and 1.2 pointed out application opportunities of transport models in the planning process. When appropriate and meaningful, transport models can make a strong contribution to selecting the most efficient and effective schemes for infrastructure, policy, and systems. Beyond the macro-economic approach, micro-economic entities can also benefit from the agent-based approach. As an example, given the modelling results, shops may adjust their opening hours or extend their facilities in order to increase profits (see Ciari et al., 2008). MATSim is hence a powerful tool to investigate both macro and micro-economic aspects of life.
1.6 Structure of the thesis

Chapter 2 introduces the classic 4-step transport model and motivates activity-based travel demand modelling. Core principles of activity-based travel demand models are outlined and most relevant implementations highlighted. Emphasis is put on the integration of activity-based travel demand models with assignment models. Chapter 2 finally introduces MATSim including core principles and its current development status.

Chapters 3 through 5 present design and implementation of the scheduling algorithms:

- Chapter 3 deals with the PlanomatX algorithm. PlanomatX is a MATSim replanning strategy producing comprehensively optimized schedules, i.e. optimized combinations of a schedule’s activity chain (number, type and sequence of activities), activity timings, and the location, mode and route choices. Some first test results of the PlanomatX algorithm are provided, with particular focus on the comparison of its performance against existing MATSim replanning strategies.

- In chapter 4 the TimeModeChoicer algorithm is presented. This algorithm optimizes activity timings and mode choices of a schedule. It can be used both as a sub-algorithm of the PlanomatX algorithm and as a stand-alone MATSim replanning strategy. Extensive test results are presented with respect to the stand-alone TimeModeChoicer, as well as to the TimeModeChoicer in combination with the PlanomatX algorithm.

- Chapter 5 illustrates the concept of schedule recycling. Aiming for reduced runtimes, schedule recycling helps avoid running the PlanomatX algorithm for each person individually. It enables re-using optimized schedules from some agents for other agents whose schedules have not yet been optimized.

Chapter 6 deals with specification and empirical estimation of the new utility function. It describes the estimation methodology applied and presents results. It explains the need for further manual calibration and discusses the new utility function’s characteristics.

Chapter 7 concludes the thesis. It summarizes the main findings and gives recommendations for further research.
Chapter 2

Theoretic background and literature review

2.1 Classic 4-step transport model

The classic 4-step transport model produces a representation of traffic flows for a geographic area over a period of time. This model emerged in the 1950s and was often referred to as UTMS (Urban Transport Modelling System). It features four steps (see figure 2.1) and aggregates space, time and travellers to reduce computational complexity:

- Spatial aggregation is achieved by zoning the study area under the assumption that all travel from and to a zone originates and ends at the zone’s centroid rather than at the actual departure and arrival locations.

- Aggregation of time is reflected by assuming static average figures throughout the modelled period (e.g., a day).

- Finally, travellers are aggregated, not taken into account individually. Flows of travellers are modelled instead. This means that all travellers (from an origin to a destination zone, using the same mode) are summed up and modelled uniformly.

Required inputs to the classic 4-step approach are socio-economic data on (expected) population and employment in the study area and data on the (expected) transport network and its service attributes. The output of the classic 4-step approach is a prediction of modal flows on the network links.
Figure 2.1: Classic 4-step model structure

Source: after Meyer and Miller (2001)

2.1.1 Overview of the four steps

The following sections provide a condensed overview of the four steps. The interested reader is referred to e.g., Schnabel and Lohse (1997), Ortúzar and Willumsen (2001), or Meyer and Miller (2001) for comprehensive reviews.

2.1.1.1 Trip generation

Trip generation predicts the number of trips emanating from and ending at each zone. Typical variables explaining trip numbers are a zone’s socio-economic attributes such as household structures, incomes, car availabilities, residential and employment density, etc. Trip generation does not take trip attributes (e.g., trip length or travel time) into account for the calculation of trip numbers.

2.1.1.2 Trip distribution

Trip distribution adds destination and origin zones to the trips that emanate from, or end at, each zone. Its outcome is an origin-destination-matrix (OD-matrix), indicating how many trips lead from a zone \( i \) to a zone \( j \), for all combinations of \( i \) and \( j \). Common methodologies for trip distribution are growth-factor and gravity models. Growth-factor models are adequate to
extrapolate an existing OD-matrix. Gravity models are inspired by Newton’s law of gravity and require no existing OD-matrix. The number of trips between an OD-pair is determined against the distance or expected travel time/cost between the OD-pair. Influence (= gravity) of zone and expected travel times/cost follows a deterrence function that reproduces the disincentive to travel as the distance or cost of travelling increases.

2.1.1.3 Mode choice

The mode choice step assigns the overall number of trips of each OD-pair to the available transport modes (e.g., car, bike, walk, bus, train/tram). Common methodologies of mode choice are again gravity models as well as discrete choice models. Gravity models belong to the group of trip-interchange models (as opposed to the out-dated, pure trip-end models), and base their outcomes on trip attributes between the respective origin and destination zones, e.g., approximated travel time/cost or just crow-fly distance. Discrete choice models may consider also zone-specific attributes such as household incomes, car ownership, etc. For both model types, trip distribution and mode choice may be derived jointly.

Mode choice is the last step in determining travel demand in the classic 4-step transport model. The resulting mode-specific OD-matrices specify how many travellers wish to travel from one zone to each other zone, by which mode.

2.1.1.4 Assignment

The assignment step aims at allocating the travel demand given by the OD-matrices to the transport supply network. Key objectives of the assignment step include the determination of

- realistic flow figures for core links, particularly for those vulnerable to congestion,
- estimates of travel times and costs between OD-pairs,
- routes between OD-pairs.

Different types of assignment models have been developed (see table 2.1). They are commonly classified by the type of route choice (deterministic or stochastic) and the type of capacity constraints (congested or uncongested):

- Route choice: A traveller in a transport network may be in the position to chose from several routes from his origin to his destination zone; for instance, various combinations of streets or different bus lines. Route choice deals with assigning specific routes to each
Table 2.1: Classification of assignment models

<table>
<thead>
<tr>
<th></th>
<th>Deterministic route choice model</th>
<th>Stochastic route choice model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uncongested</strong></td>
<td>All-or-nothing network loading</td>
<td>Stochastic network loading</td>
</tr>
<tr>
<td>transport network</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Congested</strong></td>
<td>Deterministic user equilibrium</td>
<td>Stochastic user equilibrium</td>
</tr>
<tr>
<td>transport network</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OD-pair. Deterministic route choice models select the same route(s) for all members of a class of travellers.

Stochastic route choice models come up with a probability distribution and distribute all travellers of an OD-pair accordingly. One difficulty here is that the set of all feasible routes between an OD-pair can be fairly large, but most likely includes many improbable routes. Stochastic route choice models, therefore, require first generating a reduced choice set, and the probability distribution is then applied to it. If stochastics in the transport supply network are explicitly modelled, variability in travellers’ route choice behaviour can also be modelled through, e.g., en-route choice behaviour, as opposed to the general pre-trip choice behaviour (see e.g., Hickman and Bernstein [1997], Lam and Bell [2003]).

- **Capacity constraints**: Assuming a transport system without capacity constraints, the trip numbers from the OD-matrices can just be loaded onto the network. For deterministic route choice models, this is called “all-or-nothing network loading”, since all trips of each OD-pair are fully loaded onto the one chosen route. For stochastic route choice models, the OD-pair is distributed according to the probability distribution, referred to as “stochastic network loading”.

When considering capacity constraints the traveller flow may exceed the free flow capacity limits of some network links. What results is congestion and, thus, increased travel times. In this case, travel times after the network loading are different from the empty network state that supported the route choice model. The route choice model can now identify routes other than the optimal original. “Congested” assignment models, therefore, iterate the network loading step until an equilibrium state has been found, where changing a route will no longer yield a utility gain to any traveller. This is referred to as Wardrop’s first equilibrium (Wardrop [1952]). Depending on the underlying route choice

---

1See e.g., Hoogendoorn-Lanser [2005] for concepts of route choice set generation.
models, these assignment models are classified as deterministic or stochastic user equilibrium models (see e.g., LeBlanc et al. [1975], Daganzo and Sheffi [1977]).

2.1.2 Disaggregate 4-step transport models

Classic 4-step transport models aggregate data. Above all, they lack a time dimension, making an effective analysis of traffic flows over time infeasible (e.g., analysing the dynamics of congestion). Recent models have, therefore, integrated the time dimension, particularly schedule-based public transport models. The modelling period (e.g., a day) is discretized into a certain number of time slices (e.g., 15-minute time slices) and the overall travel demand broken down into the different time slices. Within each time slice, travellers’ desired departure times can be assumed to occur at a single point of time (e.g., the time slice’s mean, see Nuzzolo and Crisalli [2004]), or uniformly distributed (Nielsen and Frederiksen [2006]). Corresponding assignment models are dynamic simulations capable of determining time-dependent user equilibria (see e.g., Wilson and Nuzzolo [2004] for a good overview).

Another attempt to disaggregate data within the 4-step model is to differentiate groups of travellers or model each traveller individually. Let us recall that the classic approach assumes one traveller type (i.e. one identical set of utility function coefficients for all travellers), where variability in the travel behaviour may be achieved only if the assignment model is stochastic. Recent models introduce several traveller groups or adopt a fully-fledged microsimulation approach where each traveller is modelled individually. Examples of the former are, e.g., Nielsen (2004) differentiating commuters, business travellers and travellers on educational/private purposes, or Nuzzolo and Crisalli (2004) differentiating frequent and occasional travellers. An example of the latter is, e.g., Vovsha et al. (2002).

2.1.3 Shortcomings of the classic 4-step transport model

The classic 4-step transport model was developed in the 1950s when “the expansion of transport infrastructure was of primary concern” (Kitamura, 1996), focusing on the identification of additional road needs to relieve congestion and increase accessibility of sub-urban developments. Given this focus and the limited and costly range of computational power, the classic 4-step transport model provided a good trade-off. The aggregation of data (space, time, travellers) and the sequential procession along four steps (limiting the conceptual complexity within each step) reduced computational requirements, yet ensured fair and satisfactory results effective in solving the defined problems.

Today, modelling questions are more complex and expansion of transport infrastructure is no longer the sole argument in transport planning. Modern travel demand management focuses on
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling 2010

more efficient use of the existing transport infrastructure. It develops and implements strategies and policies to reduce car traffic or to re-distribute it over time and space (Department for Transport [2009]). Such strategies and policies are, for instance, road pricing, single-driver charges, mode-shift incentives, intelligent transport systems,[2] etc. (Bowman and Ben-Akiva 1996). The analysis of greenhouse emissions from transport has also become important (Pas 1996; Kitamura 1996). In a first response, researchers have enhanced the classic 4-step approach by integrating the time dimension and considering groups of travellers or individual travellers (see previous section). Nevertheless, the classic 4-step approach may not serve new research goals well. Two significant problems remain (after Kitamura 1996):

- **Lack of behavioural basis:** Travel demand is determined, in the classic 4-step process, through trip generation, distribution and mode choice. However, this is not consistent with how most travellers behave. Rather than thinking about how many trips to make, a traveller would plan what needs to be done during the day, how long it will take, and where events may take place. Only then would the traveller arrange necessary trips by defining how to best get from one location to the other. The discrepancy between the classic 4-step approach and travellers’ planning behaviour becomes evident when reflecting on a typical strategy from travel demand management like road pricing. Let us assume a worker drives from his suburban home to his city work place in the morning, drives back home for lunch, drives to work in the afternoon again, and finally returns home in the evening. When a road pricing scheme is introduced, the worker may drop lunchtime’s work-home-work tour for cost reasons. He may replace it by lunching at some place in the city nearby. He may pay more for the lunch itself, but cuts cost overall by avoiding road pricing. The classic trip generation step would not capture this change in trip numbers, as it is typically not sensitive to trip attributes, such as road pricing, but only to socio-economic attributes of the origin and destination zones.

- **Trip-based model structure:** As “trip generation” and “trip distribution” indicate, the classic 4-step approach is fully trip-based. This is a problem in two respects; first, a trip may relate to another trip when travellers have to use the same mode for both trips (e.g., home-work-home by car). Hence, classic 4-step models tend to over-estimate modal shift reactions to policy changes since the mode choice step takes into account zone and/or path attributes, but not inertia from trip inter-dependencies. Second, the departure time of a trip may depend on the arrival time of another trip (e.g., home-work arrival and work-home departure). If a work commuter delays his morning trip due to a policy change he also has to delay his evening trip to complete the full work shift. It is also important to ensure that a return trip is modelled only after the outward trip (e.g., shopping-home only after home-shopping). Classic 4-step models, even those considering the time dimension, would not be able to handle such travel time dependencies.

See e.g., “Intelligent Transportation Systems Society” (2009).
2.2 Activity-based travel demand modelling

2.2.1 Core principles and definitions

Activity-based travel demand modelling emphasizes travellers’ participation in out-of-home activities as a source for travel demand. The idea that travel is an induced demand has been well accepted by researchers since it was first articulated by Oi and Shuldiner (1962), implying that what one should study first is not travel demand per se, but rather travellers’ participation in out-of-home activities that generates travel demand (Meyer and Miller, 2001). Chapin (1974) hypothesized that activity demand is motivated by basic human desires, e.g., survival, social encounters, and ego gratification, while Hägerstrand (1970) theorised that activity options available to individuals are defined by three types of constraints:

- **Coupling constraints** require other resources to be able to perform an activity (e.g., the presence of another person).

- **Authority constraints** are exogenous factors such as shop opening hours, traffic rules, etc.

- **Capability constraints** are physical or technological limitations. They are specified by time-space prisms: individuals live in a time-space continuum and, while moving in this continuum, must obey time and cost constraints on movement (see figure 2.2).
2.2.2 Advantages

Hägerstrand’s constraints of activity participation confirm that an activity-based travel demand theory is capable of relieving the shortcomings of the classic 4-step model. Coupling and authority constraints address the problem of an inadequate behavioural basis. Capability constraint addresses the trip-based model structure. The activity-based approach is thus suitable to support travel demand management analysis requirements. Particular advantages of the activity-based travel demand approach include (Meyer and Miller 2001):

- Improved capability to model non-work, non-peak travel involving complex interactions among household members, activities/trip purposes timing, resource allocation, etc.
- Improved capability to deal with trade-offs between in-home and out-of-home activities.
- Greater potential to move beyond traditional explanatory variables (i.e. zone-based socio-economics, travel time and cost).
- Greater potential to deal with the effects of household interaction, age, lifestyles, etc. on travel behaviour.

Activity-based models are foundations for specifically incorporating actions leading to travel demand (Borgers et al. 1997). Pas (1996) therefore state that “… the activity-based approach to travel demand forecasting can be considered the only real scientific revolution […] in the history of the development of travel demand forecasting models”. In comparison to activity-based developments, “[the] shift from aggregate to disaggregate models [see section 2.1.2] was only a shift in statistical technique […] and thus can be considered an incremental change rather than a shift in paradigm”.

2.2.3 Fundamental problem of activity scheduling

The fundamental problem of the models is the combinatorial size of the activity scheduling solution space (Bowman and Ben-Akiva 1996). The decision process of a traveller may have an extremely large number of feasible outcomes, meaning that the problem is not solvable by complete enumeration. Table 2.2 shows the potential size of the combinatorial problem assuming example values. Travellers themselves do not perceive the immense magnitude of solution space, since they consider only a few discrete alternatives. The modeller’s task is to develop concepts capable of solving the problem in a satisfactory time scale, and in a way that matches travellers’ behaviour.
Table 2.2: An example number of schedule alternatives an individual may face

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Number of alternatives per activity agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities (e.g., per day)</td>
<td>10</td>
</tr>
<tr>
<td>Sequencing alternatives</td>
<td>10</td>
</tr>
<tr>
<td>Timing alternatives</td>
<td>10!</td>
</tr>
<tr>
<td>Location alternatives</td>
<td>100</td>
</tr>
<tr>
<td>Mode choice alternatives (without tour constraints)</td>
<td>50</td>
</tr>
<tr>
<td>Route choice alternatives</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>10</strong></td>
</tr>
</tbody>
</table>

Source: Bowman and Ben-Akiva (1996) with modifications

2.2.4 Existing modelling concepts

The theoretic background of activity-based travel demand modelling dates back to the 1960s and ’70s (Oi and Shuldiner, 1962; Hägerstrand, 1970; Chapin, 1974). Many of the early efforts addressed the desire to better understand travel behaviour as a science, not as a planning tool (Kitamura, 1988; Pas, 1996). Research in activity-based travel demand modelling as a planning tool, directed towards prediction and forecasting, gained importance in the 1990s when the switch from infrastructure expansion to travel demand management policies took place.

Given the demand from practitioners, broader input data became available (Pas, 1996). Census and survey designs moved away from a pure travel focus (“where did you travel?”) to an activities-inclusive structure (“what did you do and how did you travel there?”). In the 1990s, computer technology had advanced to a stage where storage and processing of large data volumes was feasible and affordable - a ground-breaking enabler to the more complex activity-based approach.

Recker et al. (1986a), Kitamura (1988), and Axhausen and Gärling (1992) provided extensive overviews of the pioneering work in activity-based travel behaviour analysis. Ettema and Timmermans (1997), Goulias (2002), Timmermans (2001, 2003), and Bradley and Bowman (2006) reviewed recent developments in activity-based travel demand modelling. The following sections summarize the most important classes of activity-based travel demand models (see table 2.3) and highlight some major model implementations. To simplify, we will use the following definitions in the rest of the thesis:

---

3Several authors (e.g., Pas, 1996; Timmermans, 2003; Shiftan et al., 2007; Weiner, 2008) point out the contribution of the Clean Air Act Amendments of 1990 and the Intermodal Surface Transportation Efficiency Act of 1991 in the US.
• **Activity**: An activity is a “continuous interaction with the physical environment, a service or person, within the same socio-spatial environment [...]” (Axhausen, 2000). It is defined in purpose, timing, duration, and location.

• **Activity type**: Activity purposes can be clustered into coherent activity types (e.g., being at home, working, shopping, etc.).

• **Activity agenda**: The activity agenda is the unordered list of activity types a person performs over a certain period of time (e.g., 2×home, work, shopping).

• **Activity chain**: The activity chain is a person’s activity agenda where the sequence of participating in each activity type of the activity agenda is defined (e.g., home-work-shopping-home).

• **Schedule**: The schedule is a person’s activity chain where each activity is also labelled with information on its timing, duration, location and access mode (Axhausen, 2000).

• **Scheduling**: Scheduling is the process of generating a schedule.

### 2.2.4.1 Disaggregate activity-based travel demand models

Disaggregate activity-based travel demand models treat each traveller individually, and are based on two general steps (compare with Balmer, 2007):

• **Generation of a synthetic population**: Disaggregate models require a synthetic population of individual agents. A synthetic population is a realization of the census or survey data of the region to be modelled. It is not necessarily an exact match of the real-world population, but maintains and mirrors the socio-economic structure of the population. A census taken from the synthetic population should mirror the structure of the original census.

• **Generation of schedules**: For each individual of the synthetic population, an activity-travel schedule is developed. The schedules include activities to be performed by the individual during the modelling period, their timings and locations, and the mode of travel between the activity locations. Some models do not generate full schedules, but instead focus on partial elements of a schedule, e.g., mode choice.

---

4 See also Axhausen (1997).
5 Also just called “home”.
6 Activity chain, activity pattern and activity sequence are synonyms.
7 Schedule and (day) plan are synonyms.
8 Some models may also integrate travel routes into the schedules.
Table 2.3: Classification of existing activity-based models with a selection of relevant publications

<table>
<thead>
<tr>
<th>Model class</th>
<th>Described in section</th>
<th>Model name / Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Disaggregate</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utility-based models</td>
<td>2.2.5.3</td>
<td>Ruiter and Ben-Akiva (1978) Bowman and Ben-Akiva (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bhat (1997)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bhat (1998)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Srivinisan and Athuru (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Srivinisan and Bhat (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bradley and Vovsha (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gliere and Koppelman (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Habib and Miller (2008)</td>
</tr>
<tr>
<td>Utility-based</td>
<td>2.2.5.4</td>
<td>STARCHILD, Recker et al. (1986a,b)</td>
</tr>
<tr>
<td>micro-simulations</td>
<td></td>
<td>ORIENT/RV, Axhausen (1988)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ashiru et al. (2004)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCATS, Pendyala et al. (2004)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CEMDAP, Bhat et al. (2004)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSIMS, Hobeika (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Charypar and Nagel (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meister et al. (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MATSim, Balmer (2007)</td>
</tr>
<tr>
<td>Computational</td>
<td>2.2.5.5</td>
<td>PESASP, Lenntorp (1976)</td>
</tr>
<tr>
<td>process models</td>
<td></td>
<td>CARLA, Jones et al. (1983)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCHEDULER, Golledge et al. (1994)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMOS, Kitamura and Fuji (1998)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALBATROSS, Arentze and Timmermans (2004)</td>
</tr>
<tr>
<td>Other models</td>
<td>2.2.5.6</td>
<td>HAPP, Recker (1995)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HARP, Gan and Recker (2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AURORA, Joh (2004)</td>
</tr>
<tr>
<td>Aggregate</td>
<td>2.2.5.8</td>
<td>VISEM, Fellendorf et al. (2000)</td>
</tr>
<tr>
<td>models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhanced 4-step</td>
<td></td>
<td>VISEM, Fellendorf et al. (2000)</td>
</tr>
<tr>
<td>models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structural equations</td>
<td>2.2.5.9</td>
<td>Golob et al. (1995)</td>
</tr>
<tr>
<td>models</td>
<td></td>
<td>Golob and McNally (1996)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kitamura (1996)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Golob (2002)</td>
</tr>
</tbody>
</table>

Sections 2.2.4.2 through 2.2.4.6 will discuss existing disaggregate models, focusing on generation of schedules.

### 2.2.4.2 Utility-based models

Utility-based models follow the theory of utility-maximizing consumer choice. In classic transport modelling, only travel attributes are associated with a utility value (e.g., travel time, number of changes in public transport, etc.). In activity-based travel demand modelling, perfor-

---

9Synthetic population generation is a primarily statistical task (see, for instance, Ciari et al. 2007).
mance of activities is also associated with a utility value (see [Becker, 1965] for the pioneering model). Utility-based models replicate individuals’ decision-making processes through a two-stage choice framework (see figure 2.3). In the first stage, a choice set is generated consisting of the overall solution space or a sub-set. In the second stage, one best alternative is selected from the choice set. [Bowman and Ben-Akiva, 1996] suggested the differentiation of utility-based models by econometric models (see section 2.2.4.3) and microsimulation models (see section 2.2.4.4):

### 2.2.4.3 Econometric models

Econometric models belong to the class of utility-based models and “... use systems of equations to capture relationships among attributes” ([Bhat et al., 2004](#)). Attributes typically comprise socio-economics, activity participation and travel. Most econometric models are disaggregate and yield probabilities of decision outcomes. They place little emphasis on the first stage of choice framework (see figure 2.3). They generally produce a choice set that either assumes that travellers consider the whole solution space (= all feasible alternatives) or applies a simple search rule resulting in a rather large choice set ([Bowman and Ben-Akiva, 1996](#)). The probability of being chosen as best alternative is then determined for each alternative of the choice set, usually by discrete choice models (e.g., multinomial logit or nested logit models). The probability distribution may be translated into a specific solution alternative through Monte-Carlo simulation.

Some important contributions of activity-based econometric models are as follows:

- The first model of this class was probably a trip-based econometric model for the San Francisco Bay Area, developed by [Ruiter and Ben-Akiva, 1978](#). The model integrated...
both household-related long-term mobility/lifestyle decisions (e.g., car ownership) and short-term activity/travel decisions (e.g., activity engagement and travel mode). Variables were estimated in three sequential multinomial logit models linked through conditionality and expected utility.

- **Bowman and Ben-Akiva (2001)** presented an all-day schedule-based nested logit model structured into five tiers: (a) top tier's activity pattern model is a nested logit model with a “home” and an “out-of-home” nest. Lower tiers are (b) choice of primary tour time of day, (c) primary destination and the corresponding mode choice, (d) secondary tour time of day, and (e) secondary tour destination and the corresponding mode choice. A model prototype was applied to the Boston area, and a later production version implemented for the Portland, Oregon area.

- Bhat presents various econometric models addressing partial problems of activity-based travel demand modelling.
  - In **Bhat (1997)**, a model was developed to jointly solve the work travel mode choice and the number of non-work commute stops during the work tour. The model formulation takes advantage of both an unordered and ordered multinomial choice approach.
  - In **Bhat (1998)**, he then developed a discrete/continuous econometric model of post home-arrival activity participation behaviour. Activity type choice is the (multinomial) discrete choice. Home-stay duration and out-of home activity duration are continuous decisions.

Both models were applied to the Boston Metropolitan area.

The journal of *Transportation* dedicated a special issue to modelling intra-household interactions and group decision making (**Bhat and Pendyala, 2005**). Four of the five presented articles dealt with econometric models:

- **Srinivasan and Athuru (2005)** examined maintenance activity allocation and participation of household members. Questions concentrate on whether an activity is pursued alone, or with another household member. If the former holds true, the model then also specifies who performs the activity. The model is based on a nested mixed logit modelling framework.

- **Srinivasan and Bhat (2005)** focused on the role of intra-household interactions and their effect on time household members spend in in-home and out-of-home maintenance activities. The model specifies the duration of in-home activities as well as the decision to undertake out-of-home maintenance activities, the person conducting them and the duration. Srinivasan and Bhat use a joint mixed-logit hazard duration model.
• Bradley and Vovsha (2005) considered not only maintenance activities, but also examine the choice of entire daily activity chains. Their model consists of three layers of logit models, with the top layer dealing with the choice of overall daily activity pattern type. The middle layer considers joint activity and travel participation. The bottom layer treats generation and allocation of maintenance activities among the household members.

• Gliebe and Koppelman (2005) were also concerned with entire daily activity chains, but model only two-person households. The econometric model applied is a parallel constrained choice logit model. While the two household members choose one of several possible spatial interaction patterns they also choose an individual full-day tour pattern.

Finally, Habib and Miller (2008) presented an econometric model not based on logit models, instead using the Kuhn-Tucker optimality conditions to derive an optimum set of activities and their timings. The model deals only with workers, but is day-to-day dynamic, i.e. explicitly modelling variations between days. It abstracts “non-critical” activity types through composite activities with an average utility contribution. Location choice and travel times are considered through calibrated random variables, but not explicitly modelled. As opposed to logit models, a specific alternative is derived as solution.

Econometric models have the advantage of being based on a well-established statistical methodology and economic theory. However, they quickly tend to become very complex and difficult in respect to estimation and operationalization. Modellers are then required to apply levers like limiting the level of detail of the model outcome (e.g., low temporal granularity, implicit location choice, etc.), or neglecting temporal, institutional, or spatial constraints (Joh 2004).

2.2.4.4 Utility-based microsimulations

Utility-based microsimulation apply a sequential decision-making process in which they iterate the two stages of the choice framework (see figure 2.3). The main focus is put on the first stage, employing “... a complex search heuristic which yields a very small choice set” (Bowman and Ben-Akiva 1996). In the second stage, utility-based microsimulations typically enumerate the alternatives of the choice set and select the best alternative, e.g., through discrete choice models. The procedure may be repeated until all steps of the decision process have been completed. Rather than a probability distribution, the result is always a specific solution alternative.

---

10 No interaction” is a possible spatial interaction pattern, too.
11 The prefix “micro” refers to the simulation being based upon a disaggregate, or micro, level of individual decision-makers (Miller 2002).
STARCHILD (Recker et al., 1986a,b) was the earliest example of an activity-based microsimulation. Starting from a full enumeration of feasible schedules that take into account household interaction, STARCHILD groups similar schedules, subject to a similarity measure, and selects one distinct and representative alternative from each group. It then generates a choice set by excluding all inferior alternatives using some heuristic decision rules. The best solution from the choice set is chosen through a multinomial logit (MNL) model.

ORIENT/RV (Axhausen, 1988) is another early model, based on previous work by Poeck and Zumkeller (1976). To generate schedules, it first defines activity chain probabilities for 13 groups of behaviourally similar persons. It then assigns each person an activity chain according to the person’s group probabilities, and conducts mode and destination choice, including parking choice. The model uses logit models and determines specific solution alternatives through Monte-Carlo simulation. Travel times are calculated using a disaggregate, event-based traffic flow simulation.

From the 1990’s, a large number of utility-based microsimulations has been designed and implemented for planning agencies in the United States (Bradley and Bowman, 2006). Some relevant models are:

- PCATS (Kitamura, 1996; Kitamura and Fujii, 1998; Pendyala et al., 2004) is a model that integrates utility maximisation framework with Hägerstrand’s concept of time-space prisms to generate activity schedules. The first simulation step divides the simulation period (e.g., a day) into blocked and open periods. Prism constraints are deduced from the timings and locations of those activities that fall into the blocked periods (skeleton activities). Following simulation steps add non-skeleton activities one by one (including their timings, locations, and travel modes) until the day is complete. Nested logit models are used for the choice of activity type, location and mode choice. Hazard-based models are used for the choice of activity timings. Together with a synthetic population generator, PCATS is part of FAMOS (Pendyala et al., 2004).

- CEMDAP (Bhat et al., 2004) builds upon Bhat’s experience with econometric models (see previous section). CEMDAP features two major modules: The generation-allocation model creates individual activity agendas, taking into account household interaction. The simulation steps are whether and which household members go to work, whether they undertake shopping, and whether they undertake activities for personal business, social recreation, or other reasons. The individual-based scheduling model transforms the ac-

---

12 Further models belonging to this group are, for instance, METRO, Portland/Oregon (Bowman et al., 1999); SFCTA, San Francisco bay area (Jonnalagadda et al., 2001); NYMTC, New York (Vosha et al., 2002); MORPC, Columbus/Mid-Ohio (Anderson, 2006); ARC, Atlanta (ARC, 2009).

13 Prism-Constrained Activity-Travel Simulator

14 Florida Activity Mobility Simulator

15 Comprehensive Econometric Microsimulator for Daily Activity-Travel Patterns
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling 2010

tivity chains into schedules. It determines an optimal activity chain (commute attributes for workers and number of further tours for both workers and non-workers), makes tour decisions (mode choice, number of activities, tour duration, home-stay duration before tour), and specifies the activity characteristics (activity type, duration, location and estimated travel time to location). A multitude of utility and non-utility-based discrete choice models\textsuperscript{16} is used for the various model components (see Bhat et al., 2003). Subsequent consistency checks prevent the choice models from generating unreasonable or impossible results.

- TRANSIMS\textsuperscript{17} (Hobeika, 2005; TRANSIMS, 2009) also features two core modules but cuts the modules differently and produces more detailed end products (routes with calculated travel times, time dimension exact to the second, amongst others). The first module is an activity generator allocating, similar to PCATS, individual activity chains, taking into account household interaction. However, it also provides each individual with a preliminary schedule (activity type, location, preliminary timings, and access trip mode). The second module is a Dijkstra-based route planner calculating shortest routes between activities and updating activity timings accordingly. If the route planner cannot find routes that match the framing conditions of the person’s schedule, the schedule is fed back to the activity generator until valid routes can be found.

While the above models feature many decision steps (partially grouped into modules), Charypar and Nagel (2005) present a microsimulation model with only two decision steps. In the first step, the model assumes an arbitrary initial schedule for each individual of the population. In the second step, a Genetic Algorithm\textsuperscript{18} optimizes the schedules, subject to the underlying utility function. The Genetic Algorithm may vary all flexible schedule dimensions at a time (activity chain, tours, activity locations and timings/durations). This is different from the above models where the simulation steps sequentially deal with the flexible schedule dimensions. Meister et al. (2005) extended the model’s second step to also incorporate household interaction and mode choices.

Finally, the microsimulation model by Ashiru et al. (2004) should be mentioned. The authors emphasise the correct reproduction of individuals’ joint choice of activity start times and activity durations. Each individual is associated with an activity chain\textsuperscript{19} remaining fixed throughout the simulation. Some series of decision steps schedule activities chronologically one after the other. The underlying utility function considers time spent on activities as well as money.

\textsuperscript{16}Due to the hybrid employment of utility and non-utility based choice models, CEMDAP may also be categorized as CPM (see section 2.2.4.5).
\textsuperscript{17}Transportation Analysis and Simulation System
\textsuperscript{18}A Genetic Algorithm is a heuristic solution approach to complex combinatorial problems (see section 2.3.1.2 for a detailed description).
\textsuperscript{19}Authors do not report how they generate the activity chains. In their case study, they just assume simple “home-work-home” chains for all individuals.
earned/paid during activities.

Based on an iterative procedure and the more frequent employment of heuristics, utility-based microsimulations are capable of reducing the complexity of econometric models. [Vovsha et al. (2002)] and [Vovsha et al. (2004)] summarize the advantages of utility-based microsimulations:

- Utility-based microsimulations help reduce the computational burden of econometric models, thus avoiding the storage of large (multi-dimensional) probability matrices (see [Miller, 2002] for an illustrative example).
- They explicitly model the variability of travel demand (“emergent behaviour”) rather than probability-based average values.
- They specifically allow the inclusion of constraints during the sequential decision making process.
- They can take into account “competition” arising from constraints modelling, a good example being the decreasing attraction of over-crowded facilities.

Utility-based microsimulations and econometric models are, nonetheless, very closely linked. On one hand, several implementation examples above employ econometric discrete choice models for their step-specific choices. On the other hand, the utility function parameters underlying microsimulation search heuristics are typically estimated through econometric discrete choice models (compare Vovsha et al., 2002).

### 2.2.4.5 Computational process models (CPM)

Computational process models (CPMs) try to overcome the drawback of utility maximization models, namely that travellers do not make “optimal” decisions but rather context-dependent heuristic decisions [Joh, 2004]. Golledge et al. (1994) pointed out that CPMs have been developed to “… replace the utility maximising framework with behavioural principles of information acquisition, information representation, information processing, and decision making”. In line with this, CPMs assume a decision maker who “… tries a sequence of possibilities and selects the first one that is suitable” [Tasker and Axhausen, 1994].

CPMs are basically also microsimulations due to their disaggregate nature, the sequential decision process and the use of heuristics. However, the heuristics employed by CPMs consist of “if-then” rules, rather than utility-maximizing decision criteria. In the following, several important CPMs are presented. A more extensive discussion of CPMs can be found in e.g., Gärling et al. (1994).

---

21 Some authors do not differentiate microsimulations and CPMs at , for instance Bowman and Ben-Akiva (1996) or Kitamura (1996).
Very early modelling attempts in this line of research are PESASP (Lenntorp, 1976) and CARLA (Jones et al., 1983, page 193 ff). Both models employ rules based only on exogenous constraints. In PESASP, each individual is assigned an activity agenda where each activity type can be performed at a certain number of locations. An exhaustive enumeration of activity type sequences at the different locations is generated. Drawing on Hägerstrand’s space-time prism, the enumeration is then reduced top-down by alternatives not executable, subject to the constraints. CARLA is similar, but checks the constraints bottom-up when assembling the solution alternatives. Neither model is a genuine CPM, as no real behavioural decision heuristics are employed to select a (best) solution alternative.

SCHEDULER (Golledge et al., 1994) goes much further. From a given activity agenda, SCHEDULER builds up a schedule by chronologically choosing one activity after the other to be included into the schedule. Location choices and activity timings are determined, assuming travel times are known. The underlying decision heuristics are steered by households and individuals trying to achieve certain goals. Activities are defined as means to attain the goals; households and individuals believe and/or set priorities concerning to what (non-utility based) extent each activity can contribute to the goals (Ettema and Timmermans, 1997, pages 20–21). Scheduling decisions are subject to household interaction, previously chosen activities and exogenous constraints. Activity choice is, however, independent from reserving the possibility to perform other activities later. All rules in SCHEDULER are arbitrarily set and not estimated. In the conceptual framework, there is also some effort made to allow up-front planning of skeleton schedules and complete or re-schedule them during execution. That has, however, not been operationalized.

ALBATROSS (Arentze and Timmermans, 2004) was the most comprehensive activity-based CPM to date. It overcomes SCHEDULER’s drawback and takes opportunity of behavioral choice heuristics derived and estimated from activity travel data. Similarly to PCATS (see previous section), each schedule relies on one or more fixed skeleton activities. A set of flexible activities is drawn from individuals’ socio-economic data (household attributes, etc.). A scheduling model then hierarchically determines an all-day transport mode, the precise activity agenda, the activity chain including (rough) timing for each activity, travel tours, and location choices. Choice decisions are driven by an extensive set of constraints and, if more than one alternative remains feasible, by the above mentioned behavioral choice heuristics.

In contrast to the above models, AMOS (Pendyala et al., 1995; Kitamura, 1996; Kitamura and Fujii, 1998) is a so-called switching model. It does not generate schedules from scratch but adapts existing travellers’ schedules as reaction to policy changes. It follows four simulation steps: Given socio-economic, network and policy indicators, a stochastic response option generator develops a number of modified schedules for each individual. An activity-travel

\[^{22}\text{Therefore, they are also called “constraints-based models” (Joh, 2004).}\]

\[^{23}\text{See also “Other models: Re-scheduling models”.}\]
**pattern modifier** completes the modified schedules when they need refinement of activity resequencing, activity re-linking, mode and destination assignment, or trip timing. An **evaluation routine** assigns a utility value to each modified schedule. Finally, an **acceptance routine** evaluates the schedule utilities and decides whether one of the schedules should be accepted. If not, the model goes back to the first step. The decision criterion of the acceptance routine is a gaming theory, like balancing of achieved utility and cost of further search. AMOS has been successfully applied to the metropolitan area of Washington D.C..

### 2.2.4.6 Other disaggregate models

Some further disaggregate models exist:

- **Mathematical programming models:** Recker (1995) presents HAPP\(^{24}\) a utility-based mathematical programming model that deals with the optimization of household members’ all-day schedules. The model is a mixed integer linear programme and refers to the pick-up and delivery problem with time window constraints known from logistics. Each household member is assigned an externally specified activity agenda with locations given for each activity type. The model minimizes the disutility gained from travelling between the locations by optimizing departure times, subject to a fleet of vehicles and ride-sharing options.

- **Re-scheduling models:** HARP\(^{25}\) (Gan and Recker, 2008) extends the HAPP model by adding a module that considers travellers’ re-scheduling behaviour when faced with uncertainty. This should not be confused with switching models, such as AMOS, where travellers’ adaptation to longer-ranging policy changes are forecast. Instead, re-scheduling models assume that, at a particular time instant, re-scheduling may become necessary due to e.g., congestion, sudden external incentives/orders to include/drop activities, etc.. When this happens the HARP model checks all unfinished activities and, if necessary, re-optimizes their timings. In comparison with HAPP, the HARP utility function is extended by a variable that represents individuals’ inertia about changing plans. The variable causes the utility of a modified schedule to fall proportionally, the greater the “distance” between the original and the modified schedule.

Another re-scheduling model is AURORA (Joh, 2004). As with HARP, its aim is to model travellers’ reaction to external circumstances causing a re-schedule. AURORA adopts a utility-based microsimulation framework with eight different “operators” for the first choice stage (see choice framework in section “Utility maximization models”): duration, substitution, insertion, deletion, sequencing, location, accompanying-person, accompanying-vehicle, etc.

---

\(^{24}\)Household Activity Pattern Problem

\(^{25}\)Household Activity Re-scheduling Problem
and travel mode. Each operator applies a simple heuristic to produce one, or several, solution alternatives slightly different from the original schedule. If solution alternatives of the choice set yield a better utility than the original schedule, the alternative with the highest utility replaces the original schedule. The procedure is repeated until no better solution alternative can be found. Similar to the HARP model, AURORA takes travellers’ inertia about modifying the original schedule into account: in each iteration, an inertia index is increased by one unit that reduces the utilities of next iterations’ solution alternatives correspondingly. Thus, the more the simulation iterates and the more the solution alternatives differ from the original schedule, the higher the inertia index rises and the less probable it is a solution alternative that will exceed the utility of the currently chosen schedule.

2.2.4.7 Aggregate activity-based travel demand models

Aggregate activity-based travel demand models do not deal with individual travellers, but instead build groups of travellers whose activity-travel decisions are assumed to be homogeneous.

2.2.4.8 Enhanced 4-step models

Enhanced 4-step models basically replicate the classic 4-step approach, but enhance the steps with activity-based features. The most prominent model of this class is VISEM (Fellendorf et al., 1997), which is a commercial product (PTV, 2009). In trip generation, trip numbers are produced drawing on the probability of each zone’s population to perform certain activity chains. Trips are then distributed using a classic deterrence function that reflects a zone’s attractiveness to accommodate an activity of the activity chain. A mode choice model completes the aggregate “schedules.”26 For all steps, probabilities and parameters must be externally provided; default values exist from a survey conducted in Germany.

2.2.4.9 Structural equations models

Structural equations models are similar to econometric models in that they use systems of linear equations to capture relationships among endogenous measures of activity participation/travel and exogenous variables.28 Kitamura, 1996. Exogenous variables are considered explanatory to the endogenous measures. As opposed to econometric models, structural equations models

\[26\] Note that these “schedules” do not comply with the proper definition as they lack timing information.

\[27\] Typical endogenous attributes are e.g., number of trips, travel distance and time, time allocated to activities.

\[28\] Typical exogenous attributes are e.g., person and household attributes, network variables, or land use information.
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

do not rely on utility theory and are aggregate. More importantly, in contrast to all previously described models, they yield relationship coefficients between exogenous and endogenous attributes rather than (probabilities of) specific decision outcomes.

Golob accounts for the most extensive application of structural equations models in activity-based travel demand modelling:

- Golob et al. (1995) present a model that determines how the use of specific travel modes affects duration of out-of-home activities and travel time required. Endogenous variables include durations of various activity types and corresponding travel times. Exogenous variables include mode choice and socio-economic attributes such as gender, income, etc.

- Golob and McNally (1996) present a model to explain household activity interaction and demand for travel. Endogenous variables include durations of various activity types per gender and corresponding travel times. Exogenous variables include number of children, income and several other household attributes such as number of vehicles, driver’s licenses, etc.


Kitamura (1996) presents a model estimating the impact of travel time reduction on a rail commuter line in Japan. Individual endogenous variables include number of trips, activity durations and corresponding travel times, frequency of home-based trip chains, and total time spent at home. Exogenous variables include commute durations, work starting and end times, age, work trip mode, amongst others.

Structural equations models can illustrate the relationships among activity participation (mostly time allocation) and travel, but they cannot reflect individual decision making processes and are not appropriate in analyzing situations dependent on temporal or spatial differentiation (e.g., congestion, road pricing). Hence, one “... may not consider them truly ‘activity-based’” (Kitamura, 1996).

Therefore, structural equations models do not address the combinatorial problem formulated in section 2.2.3.

Check Golob (2009) and www.its.uci.edu/its/publications/casa.html for many further publications on structural equations models from Golob and co-authors.
2.2.5 Integration of activity-based travel demand models and assignment models

Integration of activity-based travel demand models and assignment models creates complete activity-based transport models. Such models replicate both travellers’ competition for spatial and temporal slots to realize their activity desires, and travellers’ reaction to the generalized cost of travel and activity participation in activity scheduling (Axhausen, 2006, see figure 2.4). Three mechanisms can be used to integrate activity-based travel demand models with assignment models (see figure 2.5):

- **Macrosimulation approach (“aggregate - aggregate”):** The aggregate model VISEM (Fellendorf et al., 1997) produces classic time-dependent OD-matrices as an outcome of the mode choice step. The OD-matrices can be easily fed into any deterministic or stochastic (dynamic) flow-based assignment model, similar to the classic 4-step approach. A feedback loop from the assignment model, for instance, to the mode choice step is possible at the aggregate level.

- **Hybrid approach (“disaggregate - aggregate”):** Summing up trips stored in the schedules of a comprehensive disaggregate model, the schedules can be used to produce OD-matrices. Since the schedules, by definition, include timings of activities and associated travel demand, OD-matrices can be modelled time-dependently. As for the aggregate approach, they may be fed into any deterministic or stochastic (dynamic) flow-based assignment model (see e.g., FAMOS in Pendyala et al., 2004). The drawback of this mechanism is that the individual-specific disaggregate schedule information becomes aggregate when summed up in the OD-matrices. An aggregate feedback loop from the assignment model to the activity-based travel demand model is possible, but not a disaggregate one.

- **Agent-based microsimulation approach (“disaggregate - disaggregate”):** The agent-based microsimulation approach connects travel demand models directly with assignment models. The approach keeps track of the individual throughout the whole modelling
Figure 2.5: Comparison of different approaches to integrate activity-based travel demand and assignment models

(a) Macrosimulation approach

(b) Hybrid approach

(c) (Agent-based) microsimulation approach

process, requiring that the assignment model be disaggregate and the individual traveller its basic simulation unit. The notion “agent-based” underlines that the microsimulation model explicitly considers the individuals as autonomous decision makers (“agents”). Every agent individually assesses his situation and makes decisions\textsuperscript{31} Depending on whether the activity-travel schedules already contain routes, a disaggregate route choice model must be run, to calculate the best route for each individual and trip. A dynamic traffic flow microsimulator then assigns the disaggregate travel demand onto the transport network and determines individual travel times (see e.g., ORIENT/RV in\textsuperscript{1988}Axhausen.\textsuperscript{1988}) If required or desired, a disaggregate feedback loop back to the activity-based travel demand model is feasible. Given simulated travel times and their potential differences from anticipated times, the travel demand model may decide to change an individual’s activity chain, timings of the activities within the chain, or mode choice (see e.g., TRANSIMS in\textsuperscript{2005}Hobeika or MATSim in section 2.3).

\textsuperscript{31}See e.g.,\textsuperscript{2002}Bonabeau for an introduction to agent-based modelling.
2.2.6 Discussion

Activity-based travel demand modelling is a paradigm shift. It assumes that travel is a demand induced from engaging in activities that take place at different locations. The ability to model relationship between activity participation and travel demand is a primary concern for today’s travel demand management.

However, beyond being more complex than classic models, the concepts and models presented also have drawbacks. Estimation and application of econometric models require substantial conceptual and computational efforts. Modellers tend to limit model outcome level of detail (e.g., low temporal granularity, implicit location choice, etc.), or to neglect temporal, institutional, or spatial constraints (Joh, 2004). Moreover, econometric models do not specifically mirror the decision making process. Utility-based microsimulation models can address many econometric model drawbacks, and thus, attract growing attention from the transport-planning community (Vovsha et al., 2002). CPMs try to overcome the drawback of utility maximization models: that travellers rarely make globally optimal decisions. Instead, CPMs focus on employing simple, often binary decision rules to generate schedules. This fits well with the modelling requirement to replicate people’s mechanisms of activity engagement. The heuristic rules of CPMs are, however, difficult to estimate empirically. Structural equations models are relatively easy to estimate and are able to illustrate relationships between socio-economic attributes and activity-travel behaviour. However, they are aggregate and neglect a specific treatment of activity engagement mechanisms. The same holds true for the macrosimulation model VISEM.

Three approaches of how activity-based travel demand models can be integrated with assignment models exist. The macrosimulation and hybrid approaches benefit from a simple integration with existing, aggregate equilibrium assignment models. Yet the aggregation is unsatisfactory for two reasons, particularly for the hybrid approach:

- The aggregation of activity-travel data in OD-matrices biases the temporal information of the activity-based travel demand model. The bias is most serious for a travel demand model with high temporal granularity that is fed into a static assignment model or one with only a few time slices (see section 2.1.2). The aggregation also can lose track of an individual within the assignment, most noticeably in stochastic assignment models; the assignment model equals a black box that receives disaggregate data and returns aggregate data.

- Beyond travel demand management, aggregation of data prevents an analysis of traffic emissions and other similar data. Emissions depend upon how long vehicles have been moving, or standing, respectively (“cold start” versus “warm start”). The aggregation of data in OD-matrices and the flow-based assignment makes such analysis impossible.
Agent-based microsimulation approach relieves these shortcomings. It keeps track of individuals throughout the whole modelling process, permitting very detailed analyses of model results. Vehicles’ emissions can be easily investigated, and the impact of a given policy scheme on specific sub-groups (the elderly, the suburban inhabitants, etc.) can be readily identified since “...the disaggregate model outputs can, in principle, be cut in almost any user-specified fashion” (Miller, 2002).

2.3 Introduction to MATSim

MATSim\textsuperscript{32} implements an activity-based approach to transport modelling for large samples. It is an integrated agent-based microsimulation model. The travel demand model iteratively improves schedule utility against the background of overall activity-travel costs. Travel costs are calculated using a suitable dynamic traffic flow model. In line with the agent-based microsimulation approach, MATSim requires no OD-matrices for the interface between travel demand and traffic flow model. Instead, it feeds the activity-schedules directly into the traffic flow model and keeps track of the individual throughout the whole modelling process. This procedure is similar to TRANSIMS (Hobeika, 2005). MATSim goes beyond TRANSIMS in the following aspects (partly from Balmer, 2007):

- **Concept:** MATSim offers a wide range of mutation and optimization strategies (see page 36). TRANSIMS relies on a set of statistical regression methods and discrete choice models.

- **Implementation:** MATSim’s data formats rely on the flexible XML standard (W3C, 2006). The XML data format allows a straightforward extension or cutback in data quality and quantity, if required. Furthermore, MATSim’s simulation runtime is a magnitude faster than TRANSIMS, based on efficient algorithms enabling the simulation of large-scale scenarios with several million agents.

MATSim is an open source software, jointly developed by TU Berlin and ETH Zurich. It has been applied to several scenarios such as Switzerland (7.3 million agents; see Meister et al., 2009), Berlin-Brandenburg/Germany, Toronto/Canada, Padang/Indonesia, among others. Further applications are being implemented for, e.g., Munich/Germany, Gauteng Province/South Africa, and Kyoto/Japan.

\footnote{\textsuperscript{32}Multi-Agent Transport Simulation, see www.matsim.org}
2.3.1 MATSim structure and constraints

MATSim splits into two core elements.

- The *generation of the initial demand* is the “0th iteration” of the travel demand model.

- The *demand optimization* then iterates the traffic flow and the travel demand model employing a co-evolutionary learning process. The learning process stops when none of the agents can further improve their schedules, or at a given stop criterion.

Figure 2.6 picks up the general structure of the agent-based microsimulation approach from section 2.2.5 and connects it with MATSim’s module architecture. Several assumptions and constraints underly MATSim:

- **Modelling period**: MATSim’s models fixed time periods, generally a day or any fraction of it. MATSim is not time path oriented.

- **Primary and secondary activities**: Activities that must be performed by an agent during the modelling period are assumed primary. Typically, home activities must be a schedule’s first and last activity. Beyond the home activity, each agent can be assigned further primary activities, e.g., work or education. All primary activities are assigned to fixed locations. The participation in secondary activities is, in contrast, facultative.
• **Opening hours**: Facilities can be associated with opening hours; for instance, shops are open from 10am to 8pm.

The following two sections will provide an overview of MATSim’s modules and their current state of development. The interested reader is also referred to Balmer (2007) and Balmer et al. (2008a,c).

### 2.3.1.1 Generation of initial demand

Generation of initial demand aims at creating a model population provided with a complete activity-travel schedule. It is a once-off task\(^{33}\) per model scenario. In line with the general structure of disaggregate activity-based travel demand models (see section 2.2.4), MATSim adopts a two-stage procedure for the generation of the initial demand. The two stages will be illustrated on the basis of the Switzerland scenario (see also Ciari et al., 2007).

**Generation of synthetic population**

Generation of synthetic population establishes a set of agents representing the population of the area to be modelled. Agents are associated with a home location and socio-economic information. For the Switzerland scenario, the latter comprises agents’ age, gender, employment status (yes/no), nationality, income, number of persons/children in the household, car ownership, driving license ownership, and ownership of public transport seasonal tickets - all drawn from extensive census data (Ciari et al., 2007; Swiss Federal Statistical Office, 2000, 2006). When data acquisition is more complicated, extensive statistical efforts may be required and/or less socio-economic information associated with the agents (see e.g., Balmer et al., 2008a, section 3).

**Generation of initial schedules**

Generation of initial schedules provides each agent with primary activities and a preliminary first schedule. Beyond primary activities, schedules could be any random, feasible schedules (see model by Charypar and Nagel, 2005). However, the initial schedules should be as close as possible to the final ones to save simulation runtime. Thus, several (statistical) methods are applied (Ciari et al., 2007):

• Activity chains are assigned using an Iterative Proportional Fitting Technique (see e.g., Frick and Axhausen, 2004).

\(^{33}\)Updates over time are obviously required to keep the generated initial demand consistent with the latest survey data.
• Primary activities and their locations are derived from census-based work and education commuter matrices (Vrtic et al., 2005).

• Secondary activities and their locations are determined through a neighbourhood search.

• Subtour-based travel modes are chosen using a logit model.

A deterministic route choice model finalizes the initial schedules (see description of model on page 37).

2.3.1.2 Iterative demand optimization

Iterative demand optimization aims at finding an equilibrium state where none of the agents can further improve their schedules. This resembles Wardrop’s route choice equilibrium of section 2.1.1.4, but refers to all schedule elements, i.e. activity agenda, activity chain, timing of the activities, location, mode, and route choices. The optimality of an agent’s schedule - and thus the equilibrium state - depends on three factors:

• Utility function, the measure against which a schedule is evaluated.

• Environment, such as the road and public transport network, the shop opening hours, etc. (compare with Hägerstrand’s constraints from section 2.2.1).

• Behaviour of all other agents, expressed by their schedules.

The third factor requires that an equilibrium state be found respecting interaction between all agents. MATSim does so by applying a co-evolutionary learning process (see also Balmer et al., 2008c):

1. Run the traffic flow model executing agents’ selected schedules.

2. For each agent, score the selected schedule. Drop the worst schedule if the number of schedules of an agent exceeds the maximum number of schedules allowed.

3. Select one (random or best) schedule for each agent. For a fraction of the agents (e.g., 10%), duplicate and re-plan the selected schedules. Add the re-planned schedule as a new one to the agent’s set of schedules and mark it as selected. Go back to step 1.

---

33 The initial schedules are agents’ selected schedules in the first iteration of the learning process.

34 See explanation at end of paragraph 2.3.1.2.
The co-evolutionary learning process stops after step 2 when a given number of maximum iterations has been reached. MATSim’s module structure is in line with the steps of the learning process:

- Step 1 is the exec module (this section, paragraph I).
- Step 2 is the scoring module (this section, paragraph II).
- Step 3 is the replanning module (this section, paragraph III).

### I) Exec module

The exec module is MATSim’s traffic flow model. It assigns travel demand (as given by the selected schedules) on the road network and updates selected schedules’ anticipated travel times with realized travel times. The equilibrium state is then achieved through the iterative execution of the travel demand and the traffic flow models, or the exec, scoring, and replanning modules, respectively.

Three implementations of the exec module exist. They all represent queuing systems where a street/link is modelled as a FIFO queue with a limited capacity:

- **MobSim** (Mobility Simulation) is a parallelized re-implementation of the time-step based traffic simulation presented by Cetin (2005). It is implemented in Java.

- **DEQSim** (Deterministic Event-Driven Queue Simulation) is an event-driven queue simulation (Charypar et al., 2007). It does not define a system state for each second of the simulation, but models only those time steps where an event occurs and “something happens”. This allows for a substantial simulation speed-up. However, the simulation is implemented in C++ requiring a dedicated data import and export from MATSim’s Java environment to DEQSim and vice versa.

- **JDEQSim** (Java Deterministic Event-Driven Queue Simulation) is the Java re-implementation of DEQSim and avoids the data import and export.

### II) Scoring module

The schedule execution in the exec module replaces the anticipated travel times by the realized travel times resulting from all agents’ interaction (i.e. longer travel times in case of realized, but not anticipated congestion, shorter travel times in case of anticipated, but not realized congestion). The selected schedules should now be scored to evaluate their fitness, and schedules are...
scored against the utility function. MATSim’s existing utility function resembles the following (Charypar and Nagel, 2005):

\[ U_{\text{total}} = \sum_{i=1}^{n} U_{\text{act},i} + \sum_{i=1}^{n} U_{\text{late},i} + \sum_{i=1}^{n} U_{\text{travel},i} \]  

(2.1)

where \( U_{\text{total}} \) is the total utility of the given schedule; \( n \) is the number of activities or trips; \( U_{\text{act},i} \) is the (positive) utility gained from performing activity \( i \); \( U_{\text{late},i} \) is the (negative) utility gained from arriving late at activity \( i \); and \( U_{\text{travel},i} \) is the (negative) utility gained from travelling trip \( i \). The utility itself is measured in generalized costs/earnings expressed in utility points.

\( U_{\text{act},i} \) has a logarithmic form (s.t. \( U_{\text{act},i} \geq 0 \)) inducing the marginal utility to fall the longer the activity is performed:

\[ U_{\text{act},i}(t_{\text{act},i}) = \max \left[ 0, \beta_{\text{act}} \cdot t_{i}^{*} \cdot \ln \left( \frac{t_{\text{act},i}}{t_{0,i}} \right) \right] \]  

(2.2)

where \( t_{\text{act},i} \) is the actual performed duration of the activity, \( t_{i}^{*} \) is the “typical” duration of an activity, and \( \beta_{\text{act}} \) is the marginal utility of an activity at its typical duration. \( \beta_{\text{act}} \) is identical for all activities. \( t_{0,i} \) is defined as follows:

\[ t_{0,i} = t_{i}^{*} \cdot e^{-\frac{A}{t_{i}^{*}}} \]  

(2.3)

where \( A \) is a scaling factor. \( t_{0,i} \) influences both the minimum duration and the priority of an activity. When \( t_{\text{act},i} < t_{0,i} \), \( U_{\text{act},i}(t_{\text{act},i}) \) would turn negative which is averted by the maximum term setting the utility equal to zero in this case. The smaller \( t_{i}^{*} - t_{0,i} \) (while by definition \( t_{0,i} < t_{i}^{*} \)), the steeper the ascent of the log function between \( t_{0,i} \) and \( t_{i}^{*} \) and, thus, the higher the marginal utility will be. This implies that it is favourable to shorten other activities with lower marginal utility to ensure that this activity time window can be kept. In the absence of opening hours or travel times, the maximum utility over all activities of the schedule is reached when all activities have the same marginal utility.

\( U_{\text{late},i} \) is the (negative) utility of arriving late at an activity if the activity has a latest feasible start time:

\[ U_{\text{late},i} = \beta_{\text{late}} \cdot t_{\text{late},i} \]  

(2.4)

where \( \beta_{\text{late}} \leq 0 \) is the marginal utility of being late, and \( t_{\text{late},i} \) is the start delay of activity \( i \). If an agent arrives early at an activity (before the opening time of the activity), there is no penalty, but the agent gains no utility until the activity “opens”. The activity duration continues referring to the actual arrival time, not the activity opening time.
$U_{\text{travel},i}$ is the (dis)utility of travelling:

$$U_{\text{travel},i} = \beta_{\text{travel}} \cdot t_{\text{travel},i}$$

(2.5)

where $\beta_{\text{travel}} \leq 0$ is the marginal utility of travelling, and $t_{\text{travel},i}$ is the time spent in travelling during trip $i$. The (dis)utility of travelling is not differentiated by travel modes at this time.

The utility function parameters are not empirically estimated. Instead, they have been set according the typical values of the Vickrey scenario (Arnott et al., 1993; Charypar and Nagel, 2005): $\beta_{\text{act}} = 6$ utility points/hour, $\beta_{\text{late}} = -18$ utility points/hour, and $\beta_{\text{travel}} = -$ utility points/hour. Activities’ typical durations $t^*_i$ and priorities, implicitly expressed by $A$, may differ among scenarios. Yet standard values are $t^*_i,\text{home} = 12$ hours, $t^*_i,\text{work} = 8$ hours, $t^*_i,\text{education} = 4$ or 6 hours, and $t^*_i,\text{leisure} = 4$ hours. Activities’ priorities are not differentiated.

When a schedule has been scored, the scoring module checks whether the number of schedules of an agent exceeds the maximum number allowed. If it does, the module drops the worst schedule. Thus, only the fittest schedules “survive”. The system can be considered relaxed when the spread between the best and the worst schedules has become low. This is because whatever schedule an agent would select next, the system remains robust. The maximum number of schedules allowed must be externally given and is a trade-off between computational storage capacity and system robustness. A maximum number of three to six schedules seems useful.

III) Replanning module

Agents can explore the solution space of their schedules by means of the replanning module. It allows for varying and “re-planning” their schedules in search of higher schedule utility. There are three types of replanning strategies:

- **Random mutation strategies** vary a schedule randomly. Two such strategies are currently implemented in MATSim:

  - The *Time Allocation Mutator* randomly modifies start times and durations of an agent’s schedule.
  
  - The *Trip Mode Mutator* randomly modifies the modes of all trips of an agent’s schedule. The same mode is chosen for all trips of the schedule.

The advantage of random mutation strategies is that the variation itself consumes almost no computation time. If the variation is favourable and yields a higher utility, the agent will prefer the new schedule over the old one. If not, the agent can just ignore the schedule and keep the old one until a better schedule may have been found in some later iteration.

---

37Technically, the trip mode mutator is called change leg mode. The different name in the thesis has been given for consistency reasons.
One disadvantage of random mutation strategies is the possible requirement for many iterations until the system relaxes to an equilibrium state.

- **Best response strategies** optimize a schedule, given last iteration’s travel times calculated by the exec module. Again two strategies are currently implemented in MATSim:
  - The route choice strategy optimizes the routes of a schedule. It differentiates by the mode choice: for car mode, specific routes are calculated using an enhanced time-dependent Dijkstra algorithm (Lefebvre and Balmer, 2007). For public transport mode, no specific routes are calculated, but average travel times are estimated that are twice the corresponding shortest car free-flow travel times. For bike and walk modes, average travel times are estimated based on crow-fly distances.
  - The Planomat strategy is a modified version of the GA-based travel demand model presented by Charypar and Nagel (2005) and Meister et al. (2005). It simultaneously optimizes activity timings and mode choices of a schedule. It assumes the same tasks as Time Allocation Mutator and Trip Mode Mutator, but finds an optimal rather than a random solution. Furthermore, Planomat does not just assign one mode to all trips. It optimizes the individual subtour modes according to a feasible mode chain analysis (adopted from Miller et al., 2005).

Best response strategies obviously need less iterations to relax to a system equilibrium. In general, time savings from fewer iterations outweigh increased computation time of a best response strategy (Balmer et al., 2008c). Hence, best response strategies should be preferred if computation time is a factor.

- **Constraints-based strategies** are a hybrid between the two alternatives above. They extend random mutation strategies as they obey externally given constraints, but they do not genuinely optimize a schedule. There is one such strategy implemented in MATSim:
  - The location choice strategy re-plans secondary locations (Horni et al., 2008). It does so by creating a set of feasible locations for the secondary activity in question. It then randomly selects a location from the choice set and checks whether the location, in terms of Hägerstrand’s theory of time-space-continuum (see section 2.2.1), is feasible. If it is, the location is selected as the activity’s new location. If not, the strategy iterates around refining the choice set and trying a new location.

Usually, 10% of agents are eligible in an iteration to re-plan their schedules.38

38 Note that a schedule-based public transport route choice model has just been developed (Rieser, 2010).
39 Miller et al., 2005 distinguish “return-mode” and non-return-modes”: if car or bike modes are used for a tour, the mode must be kept throughout the entire home - out-of-home - home chain. Cars/bikes have to be returned back home at the end of the day. No such constraint exists for other modes such as walking or public transport.
40 The share of agents eligible to re-plan must be provided externally. In theory, one may allow all agents to re-plan in every iteration. However, this effect would be questionable. As an example, let us just look at route...
2.3.2 Discussion

MATSim implements an activity and agent-based approach to transport modelling. Its efficient implementation is designed to favour the simulation of large-scale scenarios. Previous sections have set out MATSim’s process architecture and underlined its high level of sophistication. Most relevant potential to further improve MATSim exists in the following areas:

1. MATSim’s exec module simulates only road traffic. Its extension to a multi-modal tool is of primary concern (see Rieser 2010).

2. MATSim models “demand agents”, or travellers respectively. The agent-based approach can also be applied to the supply side, i.e. agents providing or removing capacities in networks or facilities (see first approaches in Ciari et al. 2008).

3. MATSim assumes fixed activity chains of the initial schedules throughout the iterative demand optimization. This is a legacy constraint one may want to remove. In line with the general need to keep computational performance high and the findings from section 2.3.1.2, a best response strategy may be added to the replanning module that also optimizes the activity chain of a schedule.

4. MATSim’s current utility function, as set by Charypar and Nagel (2005), is problematic, particularly because
   - current parameters of the utility function have been set according to the Vickrey scenario (Arnott et al. 1993). They have not been estimated empirically.
   - significant attributes of user behaviour are still missing, for instance the monetary cost of travelling.

This thesis will address points 3 and 4. Chapters 3 through 5 will deal with the design and implementation of efficient replanning algorithms that also incorporate the optimization of a schedule’s activity chain. Specification and empirical estimation of an improved utility function will be discussed in chapter 6.

choice and assume that all agents take route A because route B, under free flow conditions, takes slightly longer. Now we run the exec module and see that route A becomes congested and takes longer than route B when it is empty. If we give all agents the possibility to re-plan, all agents will change to route B because the exec results say it is currently faster. Running again the exec module, route B will become congested while route A reverts to the faster route, and so forth. Comparable problems in standard assignments (see section 2.1.2) are solved through a dynamic definition of the iteration’s optimal share of agents/cars/travellers allowed to re-plan (Frank-Wolfe algorithm, Frank and Wolfe 1956). However, MATSim does not apply such dynamics (yet).
Chapter 3

Comprehensive schedule optimization: PlanomatX

PlanomatX is a new MATSim replanning strategy (see figure 3.1) producing comprehensively optimized schedules, i.e. optimized combinations of a schedule’s activity chain (number, type and sequence of activities), activity timings, and location, mode and route choices.

3.1 Problem formulation

The optimality, or fitness, of a schedule is measured against its utility. Optimizing a schedule means maximizing its utility. Drawing on section 2.3.1.2, the problem of maximizing a schedule’s utility can be expressed by the following objective function:

$$\max U_{total} = \max \left[ \sum_{i=1}^{n} U_{act,i} + \sum_{i=1}^{n} U_{late,i} + \sum_{i=1}^{n} U_{travel,i} \right]$$

(3.1)

where $U_{total}$ is the total utility of the given schedule; $n$ is the number of activities/trips; $U_{act,i}$ is the (positive) utility gained from performing activity $i$; $U_{late,i}$ is the (negative) utility gained from arriving late at activity $i$; and $U_{travel,i}$ is the (negative) utility gained from travelling trip $i$.

The objective function formulates a mixed-integer, non-linear, non-convex problem. An analytical, thus quick, solution approach is unknown for this class of problems. It may thus “... not be possible to solve for an optimal solution. [...] It still is important to find a good feasible solution that is at least reasonably close to being optimal. Heuristic methods commonly are used to search for such a solution” (Hillier and Lieberman, 2005).

$^{1}$The sub-function $U_{act,i}(t_{act,i}) = \max \left[ 0, \beta_{act} \cdot t_{act,i}^{*} \cdot \ln \left( \frac{t_{act,i}}{t_{0,i}} \right) \right]$, features a non-convex solution space, as shown in Charypar and Nagel (2005).
3.2 Heuristic solution algorithm

Charypar and Nagel (2005) and Meister et al. (2005) have shown that a Genetic Algorithm (GA) can solve the utility maximization problem above. Their work has led to the implementation of MATSim's Planomat strategy (see section 2.3.1.2). Extension of the Planomat strategy toward a comprehensive schedule optimization would be a straightforward development path. Yet “GAs are known as rather inefficient” (Charypar and Nagel, 2005). They relax very significantly to a (nearly-)global optimum (e.g., Hillier and Lieberman, 2005; Hasan et al., 2000), but risk spending a large part of their search process in potentially unpromising areas of solution space. One reason is the missing cycling prevention, another the random mutation operator (Rahoual and Saad, 2006). Considering MATSim’s requirements, a perfect schedule optimization is desirable but not imperative. In fact, in many cases what people use as their schedules “... is far from being optimal” (Charypar and Nagel, 2005). The schedule optimization instead needs to be “good” but not optimal, as long as computational time is kept low. The class of Gradient Algorithms matches the above requirements. From any initial solution, the Hill Climbing Algorithm just follows the steepest gradient of objective function, stopping when no further improvement is possible. It relaxes quickly, but the solution is likely to be a local optimum. The Tabu Search Algorithm (Glover, 1989) is a more elaborate Gradient Algorithm that can overcome this drawback. It is equal to a Hill Climbing Algorithm until it has found the first (local) optimum. It may then select inferior solutions until a better solution has been found. A tabu list storing all previous iterations’ selected solutions avoids cycling. Moves
that would reach these solutions again are forbidden. A stop criterion\(^2\) lets the algorithm finish. The Tabu Search Algorithm quickly reaches “ok”-solutions followed by gradual improvement steps.

PlanomatX implements a Tabu Search heuristic. Given an arbitrary base solution, PlanomatX tries to proceed toward the steepest gradient, or more simply described, toward the steepest ascent of the utility mountain range. Considering MATSim’s scheduling problem, the utility mountain range has not only two dimensions (north-south, east-west) but five: the activity chain, the location, route and mode choices, and the activity timings. In order to master the increased number of dimensions, the PlanomatX algorithm solves the problem hierarchically applying two nested optimization loops (see figure 3.2):

- **Outer loop, steps B/C/E**: The outer loop implements the Tabu Search principle and solves for the best activity chain. The loop steers the creation of a list \(N\) of \(K\) neighbourhood activity chain solutions (number, type and order of activities), drops those neighbourhood solutions

---

\(^2\)E.g., minimum improvement over last \(n\) iterations or just overall number of iterations.
solutions that are tabu, scores the remaining neighbourhood solutions, selects the best from among them, updates the tabu list with the best solution, and sets the best solution as next iteration’s base solution.

- **Inner loop, step D**: For each created neighbourhood activity chain solution \(k\), the inner loop optimizes the lower tier decisions of location, route and mode choices as well as activity timings.

The nested loop structure implies that, when choosing the best activity chain, the outer loop actually chooses from among the best possible schedules for each activity chain. If a neighbourhood solution has a lower utility than another one, the whole lower-tiers-branch of the activity chain can be dropped. This quickly limits the solution space. Because the solution space increases exponentially with the number of attributes, this substantially limits complexity.

The algorithm stops when an externally given maximum number \(n\) of outer loops iterations has been reached or when no more non-tabu neighbourhood solutions are available. The algorithm’s optimization result is the highest-scored solution of the final tabu list. The following sections will highlight PlanomatX’s process steps in more detail.

### 3.2.1 Initialization of the algorithm (step A)

The share of agents eligible per MATSim iteration to re-plan their schedules is usually 10% (see section 2.3.1.2). For every agent replanned by the PlanomatX strategy, MATSim’s controller environment hands over a comprehensive data set containing the following information (see figure 3.3):

- **Personal information**: Personal information includes the agent’s mandatory personal ID plus optional socio-economic data, e.g., age, gender, driver’s license, car availability, etc.

- **Knowledge**: The agent’s knowledge indicates the activity types and corresponding spatial locations the agent knows. It also states whether or not an activity type is primary to the agent (thus must be performed by the agent during the day).

- **Plan**: The plan is the agent’s currently selected schedule. It provides information on the activities performed by the agent, including activity timings and activity locations. It also provides information on the trips between the activities, including travel mode and travel timings. Travel route is an optional information.

---

3Locations of activity performance are coordinates, encoded as facilities in the schedules.

4Trips are called legs in the schedules.

5Note that PlanomatX does not require the input schedule to be fully consistent or realistic. PlanomatX will only adopt the activities’ order and durations as well as trips’ mode choices. No further schedule information is maintained.
Figure 3.3: Example of an agent’s input data to PlanomatX algorithm in XML format. Note that, technically speaking, PlanomatX receives the data in Java object data format.

<person id="100" age="56">
  <knowledge>
    <activity type="home">
      <location id="12" isPrimary="yes" />
    </activity>
    <activity type="leisure">
      <location id="43" />
    </activity>
    <activity type="shopping">
      <location id="41" />
    </activity>
    <activity type="work">
      <location id="1" isPrimary="yes" />
    </activity>
  </knowledge>
  <plan score="100.21585002427452" selected="yes">
    <act type="home" link="174" facility="12" start_time="08:00:00" dur="08:00:00" end_time="08:00:00" />
    <leg mode="car" dep_time="08:00:00" trav_time="00:00:00" arr_time="08:00:00" />
    <act type="work" link="91" facility="1" start_time="08:00:00" dur="16:00:00" end_time="16:00:00" />
    <leg mode="car" dep_time="16:00:00" trav_time="00:00:00" arr_time="16:00:00" />
    <act type="shopping" link="41" facility="41" start_time="16:00:00" dur="02:00:00" end_time="18:00:00" />
    <leg mode="car" dep_time="18:00:00" trav_time="00:00:00" arr_time="18:00:00" />
    <act type="leisure" link="59" facility="43" start_time="18:00:00" dur="02:00:00" end_time="20:00:00" />
    <leg mode="car" dep_time="20:00:00" trav_time="00:00:00" arr_time="20:00:00" />
    <act type="home" link="174" facility="12" start_time="20:00:00" dur="04:00:00" end_time="24:00:00" />
  </Plan>
</person>

Initializing the calculations PlanomatX extracts the activity chain from the schedule received. For this activity chain, PlanomatX optimizes location, route and mode choices as well as activity timings (see step D, section 3.2.4, for detailed explanations). This ensures that a best schedule solution is found for the received activity chain. PlanomatX adds the schedule solution as first element to the tabu list and sets it as base solution for the neighbourhood creation (step B) of the first outer loop iteration.

3.2.2 Neighbourhood creation (step B)

PlanomatX’s neighbourhood creation deals with generating a list $N$ of $K$ neighbourhood activity chain solutions. As opposed to space or time, the activity chain dimension is discrete, not continuous. A neighbourhood solution can be created by three different moves (see figure 3.4). One can change a base solution’s overall number of activities, its sequence of activities, or the type of one or more of its activities. PlanomatX is able to perform all three types of moves. Creating a neighbourhood activity chain solution, only one move is conducted at a time. The
Figure 3.4: PlanomatX process flow with focus on neighbourhood moves

share of moves contributing to the overall number of neighbourhood solutions has to be provided externally\(^6\) (e.g., 60% changing the number of activities, 20% changing the sequence of activities, 20% changing the type of activities). Note that all moves apply only to an activity chain’s second to second-to-last activity. The first and the last activity are fixed home activities (see section 2.3.1). If the base activity chain features only one or two activities (“Home” or “Home-Home”), the share of moves is automatically adapted to 100/0/0 since, in this case, changing activities’ sequence or changing type of an activity are not valid options.

---

\(^6\) An endogenous search for the best share is unreasonable, because historic data is not appropriate to forecast probabilities of best neighbourhood moves.
3.2.2.1 Change-number sub-algorithm (step B1)

The change-number sub-algorithm creates neighbourhood activity chain solutions through insertion or removal of an activity into/from the base activity chain. The proportion of insertion/removals has to be provided externally. Unless a priori information is available, 50/50 is an obvious default share. Calling the change-number sub-algorithm, a random generator decides whether to insert or remove an activity. The random generator is set so that the insertion/removal share is matched over the $k_{\text{number}}$ neighbourhood solutions for number of activities to be changed. Figure 3.5 illustrates the process flow of the change-number sub-algorithm.

Insertion operator (step B1.1)

The insertion operator inserts an activity into a base activity chain. It is initialized by providing each “gap” of the base activity chain with a list of activity types eligible to be included. PlanomatX’s configuration offers three settings:

1. “All”: All activity types that exist in the scenario are eligible to every agent.
2. “Knowledge”: Each agent can choose only from the activity types he knows.
3. “Customized” (standard setting): User-specific rules can be established. Currently, the following rules are operational:
   - Children under 6 years can choose only from the activity types they know, usually “home” and “kindergarten”. 
Figure 3.6: Step B1.1: Insertion operator (displaying activity types only, no trips)

- Children and teenagers from 6 to 17 years may perform every activity type but “work”, “education_kindergarten”, and “education_higher”. Exceptions to the rule are known activity types. They may perform either “education_primary” or “education_secondary” but not both. The preferred option depends on their knowledge.
- Adults of 18 years or older may perform every activity type except “education_kindergarten”, “education_primary”, and “education_secondary”. Exceptions to the rule are known activity types.

The generated list of activity types eligible to be included is adopted for the first gap (between first and second activity), without modifications. For the remaining gaps, the list is reduced by the type of the activity “in front of” the gap (see figure 3.6). This avoids creation of duplicate neighbourhood activity chain solutions. Figure 3.7 provides a graphic example of the rule.

After initialization, the insertion operator inserts one activity type every time it is called. Doing so, it iterates around the gaps (see figure 3.6): it starts inserting an activity at the first gap, then at the second gap, at the third gap, etc. After having inserted an activity at the last gap, it returns to the first gap, inserting the second element of the corresponding activity type list. The number of maximum calls is limited by two constraints:

- The theoretical maximum number of combinations of gaps and activity types is $T + (T - 7)$.

Either work types “work_sector2” and “work_sector3”.

---

7 Either work types “work_sector2” and “work_sector3”.

46
1) \times (Z - 2)$, where $T$ is the number of activity types the agent knows ($= \text{length of the full list}$) and $Z$ the number of activities in the base activity chain.

- Usually, however, number of insertions is limited by the neighbourhood size allocated to the insertion operator. Drawing on previously used sample shares, this is, for instance, 3 when neighbourhood size is 10, share of the change-number operations is 60%, and share of insertion is 50%.

Duration of the inserted activity is set to 1 second (see step D2 for rationale). Start and end times, as well as the activity location, remain undefined. The trip added between the inserted activity type and the following activity is a copy of the trip connecting the activities that now surround the inserted activity. This ensures that some preliminary travel time is allocated to this trip (which is relevant to location choice; see step D1) and that a mode is preliminarily chosen (which is relevant to route choice; see step D2). Missing activity attributes, as well as the correct trip attributes, are only determined once the newly created activity chain has successfully passed the tabu check in step C. This saves runtime.

**Removal operator (step B1.2)**

The removal operator deletes an activity from the base activity chain. Like the insertion operator, it iterates around the activities of the base activity chain. When called the first time, the removal operator removes the second activity. Called the following time, it removes the third activity, and so forth.

---

8 $10 \times 0.6 \times 0.5 = 3$

9 Remember that the first and last activity of an activity chain are typically the agent’s home activity (section 2.3.1).
The theoretical maximum number of removals is $Z - 2$, where $Z$ is again the number of activities of the base activity chain. This number is considerably smaller than the maximum number of insertions (see previous paragraph). It is further reduced when we think about primary activities. Some inner activities (activities 2 to $Z - 1$) may be primary activities. The removal operator, therefore, verifies the following rules before removing an activity:

1. Check whether the activity to be removed is a primary activity. If so, proceed to rule 2. If not, remove the activity.

2. Check whether the activity chain contains the primary activity several times (e.g., work at facility x from 8 am to 1 pm and from 2 pm to 5 pm). If the primary activity exists several times, remove the item in question. If not, keep it and move on to the next activity of the activity chain. Re-start with rule 1.

The reason for rule 2 is that the insertion operator may include both primary and secondary activities. If the removal operator was forbidden to remove primary activities, such an activity - included by the insertion operator - can never be dropped again. The fact that the removal operator may drop an “original” primary activity of a certain activity type and keep a “new” primary activity of the same type instead poses no threat. This procedure is just a variant of changing the order of activities in the activity chain (see also section 3.2.2.2).

**Copy-paste of base activity chain (step B1.3)**

The two previous sections outlined the maximum number of insertion and removal operator calls. In case the allocated neighbourhood size exceeds the maximum number of calls, the surplus neighbourhood slots are filled with the unchanged base solution and marked as “not new”. Marked “not new”, the later tabu check drops the solutions from the neighbourhood list N without further examination. This saves runtime.

**3.2.2.2 Change-order sub-algorithm (step B2)**

The change-order sub-algorithm creates neighbourhood activity chain solutions by swapping two activities of the base activity chain. Only activities of different activity types may be swapped, requiring a check whether the base activity chain possesses four or more activities. If so, the sub-algorithm starts with step B2.1. If not, it goes directly to step B2.2.

---

10 Activities have no PlanomatX moving history.
11 Again, we recall that the first and the last activity must be home activities, preventing any possibility of swapping activities if the plan has three or less activities.
Figure 3.8: Step B2: Process flow of the change-order sub-algorithm

---

**Swap operator (step B2.1)**

The swap operator identifies the activities to be exchanged through two nested loops (see figure 3.9). Activity 2 is the start position for the outer loop, activity 3 for the inner loop. Every time the operator is called, it increments the position of the inner loop until the loop has reached the second-last activity of the base activity chain. Like this, all swap combinations of activity 2 can be accessed. When the inner loop has reached the second-last activity, the outer loop’s position is set to activity 3, the inner loop’s position to activity 4. Again, the inner loop’s position is incremented, and so on. When called, the operator distinguishes two cases:

1. If the two activities feature different activity types, the operator swaps them. The swap includes all attributes of the activities (type, location, timings), yet the corresponding legs remain untouched for the moment. They will only be adjusted after the activity chain has successfully passed the tabu check, saving runtime.

2. If the two activities feature equal activity types, the operator proceeds through the pairs of activities (see above inner/outer loop description) until it has found a couple for which case 1 holds true.

The theoretic maximum number of operator calls is reached when the outer loop has been set to the third-last activity. This is \((z - 3)!\) assuming case 1 always holds true.

**Copy-paste of base activity chain (step B2.2)**

For the change-order sub-algorithm, copy-pasting the base activity chain applies in two cases:

- The base activity chain has three or less activities.
Figure 3.9: Step B2.1: Activity swap operator (displaying activity types only, no trips)

- More neighbourhood slots have been allocated to the change-order sub-algorithm than new activity chain solutions can be created (compare with step B1.3)

In both cases, step B2.2 fills the neighbourhood slots with the unchanged base activity chain and marks it as “not new”.

3.2.2.3 Change-type sub-algorithm (step B3)

The change-type sub-algorithm creates neighbourhood activity chain solutions by changing the type of a base chain activity.

Change-type operator (step B3.1)

Similar to the insertion operator (step B1.1), the change-type operator is initialized by assigning each activity of the activity chain a list of activity types eligible to be exchanged against (see figure 3.11). Each list comprises all activity types the agent knows except the associated activity’s type.

After initialization, the selection of activities whose types should be modified is semi-random. Called the first time, the change-type operator randomly chooses an activity of the activity chain. From the second call onwards, it iterates around the activities, starting from the previously chosen activity. Being called, the operator distinguishes two cases:
Figure 3.10: Step B3: Process flow of the change-type sub-algorithm

Figure 3.11: Step B3.1: Change-type operator (displaying activity types only, no trips)

1. If the activity is not a primary activity or, otherwise, not the only one of that type within the activity chain (see step B1.2) the activity’s type can be changed. The operator modifies the activity type, but does not touch other attributes of the activity (timings, location). Those attributes, as well as the corresponding trips, will be adjusted once the new activity chain has successfully passed the tabu check. This saves runtime.

2. If the activity is a primary activity, and the only one of that type within the activity chain, the change-type operator proceeds iterating around the activities until it has found an activity for which case 1 holds true.
The theoretical maximum number of calls of the change-type operator is \((T - 1) \times (Z - 2)\), where \(T\) is the number of the activity types the agent knows, and \(Z\) the number of activities in the base activity chain.

**Copy-paste of base activity chain (step B3.2)**

If more neighbourhood slots have been allocated to the change-type sub-algorithm than new solutions can be created, step B3.2 fills the surplus neighbourhood slots with the base activity chain and marks them as “not new” (compare with steps B1.3 and B2.2).

### 3.2.3 Tabu check (step C)

Step C checks whether or not neighbourhood activity chain solutions created in step B are tabu. A neighbourhood solution is tabu if it matches an activity chain stored in the tabu list, i.e. if it matches one of the activity chains that has been selected as best solution in one of the previous outer loop iterations (see step E for more details). If a neighbourhood solution is tabu, it is dropped from the neighbourhood and no longer considered a valid option.

For each neighbourhood activity chain solution, the tabu check first verifies whether or not the solution is marked “not new”. If marked “not new”, step C drops the solution from the neighbourhood since the solution is the base activity chain, or the previous iteration’s best solution respectively (see step B). Otherwise, step C compares each neighbourhood solution with the tabu list entries \(n - 2\) to 0, where \(n\) is the number of the outer loop’s current iteration\(^{12}\). The backwards direction \(n - 2\) to 0 reduces the runtime since

- the comparison stops if the equality check is positive, and
- the probability of finding an equal activity chain in the tabu list is higher for the late entries than for early ones.

The equality check is based on comparison of the number and type of activities. If the number of activities in the activity chain is different, the activity chains cannot be equal. Type of activities and their order in the activity chain are compared only if the number of activities in the two activity chains is identical. If this comparison is positive, the two activity chains are considered equal.

At the end of the tabu check, PlanomatX verifies that the resulting neighbourhood is not empty. If not empty, PlanomatX continues with step D. Otherwise, it proceeds to step F.

---

\(^{12}\)Remember that tabu list entry \(n - 1\) requires no comparison since equal neighbourhood activity chain solutions would have been dropped by the “not new” check. Tabu list entry 0 is the activity chain of the original schedule entered by the controller.
3.2.4 Optimization of neighbourhood solutions (step D)

Step D is PlanomatX’s inner loop. Each neighbourhood activity chain solution that has passed the tabu check is optimized with respect to location, route, and mode choices, as well as activity timings. Step D includes providing/extendng the neighbourhood activity chains with those activity and trip attributes that may still be missing from the neighbourhood creation.

Detailed descriptions of step D will start with step D2. Step D1’s role is clearer after one has seen step D6. Step D1 will be outlined in section 3.2.4.5 together with step D6.
3.2.4.1 Location choice (step D2)

A location choice is necessary for those neighbourhood activity chain solutions that

- have seen an increase in the number of activities (step B1.1),
- have seen a change in the order of activities (step B2.1), or
- have seen a change in the type of an activity (step B3.1).

Step D2 draws on MATSim’s existing location choice strategy. The location choice strategy identifies a schedule’s primary activities, keeps them fixed and relocates intervening secondary activities obeying given time-space constraints (see section 2.3.1.2 and Horni et al., 2008). PlanomatX’s step D2 can involve the location choice strategy in two ways:

1. Full location choice: In the full location choice mode, step D2 hands the whole schedule over to the location choice strategy, which then first builds sub-chains reaching from one primary activity to the next. It then relocates all sub-chains’ inner activities, or the schedule’s secondary activities respectively. This implies that two cases can be distinguished by those activities that have been modified by one of the above neighbourhood moves, i.e. increase in number, change of order, or change of type:

   - A modified activity is primary and remains untouched by the location choice strategy. The activity can be primary if it was primary before the move (change order) or if the move made it primary (change-number, change-type). The latter can happen, by definition, when a home-activity is inserted (change-number: insertion) since a home-activity must always take place at the agent’s primary home location. The latter can also happen by coincidence when the assigned activity type and the original activity’s location result in a primary activity (change type).

   - In all other cases, a modified activity is located (change number: insertion) or relocated (change order, change type), since it is secondary.

2. Reduced location choice: In the reduced location choice mode, step D2 (re-)locates only those activities modified by one of the neighbourhood moves. To do so, step D2 needs to amend, or “fake”, the current version of the neighbourhood schedule. Vis-à-vis location choice strategy, step D2 declares the surrounding activities of the activity/ies to be located primary activities resulting in one or two “sub-chains” (see figure 3.13). Step D2 then hands over one or two sub-chains to the location choice strategy identifying feasible activity locations.
In both cases, special attention must be paid to sub-chains’ travel time budgets. A travel time budget is the sum of a sub-chain’s trip travel times. It is critical to the radius of the eclipse where the location choice strategy searches for feasible locations (see Horni et al., 2008). Prior to sending the schedule/sub-chains to the location choice strategy, PlanomatX needs to assign reasonable travel time budgets to the sub-chains to avoid the following scenarios:

- If a sub-chain’s travel time budget is (too) high, the location choice strategy may choose very distant locations, contradicting PlanomatX’s optimizer specification.

- If the travel time budget is too low, the location choice strategy may find no reachable location. In this case, the strategy launches a default location choice process in which any location is randomly selected where the activity type can be conducted. Again, this contradicts PlanomatX’s optimizer specification.

- Finally, if the travel time budget is 0 seconds, the location choice strategy does nothing and no location is assigned at all.

---

13. Remember that the current neighbourhood solution is only the hull of a schedule: an activity chain has been defined, but locations, routes, activity timings and modes may still be corrupted or undefined.

14. Except from home-activities that are set only at the agent’s primary home location; see full location choice.
3.2.4.2 Route choice (step D3)

Once the activities locations of each neighbourhood activity chain solution have been defined, best routes between the locations need to be found. Step D3 draws on MATSim’s existing route choice strategy, implying the following:

- For car trips, shortest routes are determined through a Dijkstra A-Star-Landmarks algorithm, using loaded link travel times from the last traffic flow simulation (Exec Module).
- For public transport trips, estimated travel times instead of specific routes are determined. These are twice the car travel times in a free flow road network. Those shortest free flow routes are, again, calculated through a Dijkstra or an A-Star-Landmarks algorithm.
- Like public transport trips, no specific routes are determined for walking and cycling trips. Instead, walking and cycling travel times are estimated multiplying the euclidean distance between origin and destination with an assumed average speed.

Car routes remain fixed throughout the remaining optimization process. On one side, this ignores their dependence on the departure times since they are calculated based on loaded network figures. On the other side, the bias is assumed to be small.

3.2.4.3 Optimization of activity timings and mode choice (step D4)

Step D4 is the optimization of activity timings and mode choice. Two options are available:

- Option 1 draws on MATSim’s existing Planomat strategy (see section 2.3.1.2).
- Option 2 is the newly developed TimeModeChoicer (see chapter 4 for an in-depth description).

Both options feature a sub-tour based mode choice. The difference between the options is their solution approach; Planomat implements a Genetic Algorithm while the TimeModeChoicer relies on a modified Tabu Search.

3.2.4.4 Scoring of the schedule (step D5)

After optimal activity timings and mode choices have been defined, the neighbourhood activity chain solutions become complete schedules. To compare and evaluate the fitness of each

---

15 See Miller et al. (2005): A tour is a series of consecutive trips. The origin of the first trip and the destination of the last trip must be identical. A tour may consist of several sub-tours if another tour can be identified within it. A sub-tour is thus defined as a tour within another tour.
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

schedule, they need to be scored, i.e. their utilities determined. PlanomatX draws on MATSim’s standard scoring module (see section 2.3.1.2). However, as opposed to the utility function presented in section 2.3.1.2, a new utility function needs to be implemented. Chapter 6 describes the new function, its rationale and its empirical estimation.

3.2.4.5 Storage and retrieval of optimized schedules (steps D6 and D1)

Step D comprises a large set of functionalities. In every outer loop, it is repeated for each neighbourhood solution. Section 3.3.2.3 will show that step D consumes a major part of PlanomatX’s overall runtime. As MATSim aims to be applicable to large-scale scenarios, short runtimes are a central concern. The most straightforward way to reduce any step’s runtime is to make it superfluous. Steps D1 and D6 attempt to make steps D2 to D5 superfluous as often as possible.

Storage of optimized schedules (step D6)

Step D6 stores every optimized neighbourhood schedule solution so that it can be re-used in later outer loop iterations. PlanomatX provides seven lists to store the solutions: a list for all solutions with two activities, another list for all solutions with three activities, with four activities, and so forth. The seventh list stores all solutions with eight or more activities. This storage structure helps reduce retrieval runtimes, as outlined in the next paragraph.

Retrieval of optimized schedules (step D1)

At the very beginning of step D, PlanomatX verifies whether the neighbourhood activity chain solution to be optimized has already been optimized in one of the previous outer loop iterations. This implies the following:

- The check is conducted from the second outer loop iteration onwards.
- The check verifies the length of the activity chain to be optimized and searches the corresponding list (see step D6) for schedule solutions with an equal activity chain. The storage structure using multiple lists reduces runtimes in two ways; first, the number of solutions the current activity chain solution needs to be compared with is drastically reduced. Second, as opposed to the tabu check (step C), the equality check itself can waive the comparison of the number of activities, since each two activity chains to be compared have the same length.

- Relevant solutions (that the current activity chain solution needs to be compared to) are those that have not been selected as best in one of the previous outer loop iterations. If

---

An exception is the seventh list storing all solutions with eight or more activities. For this list, the comparison of the number of activities in the activity chains is maintained.
they had been selected as best, they would be elements of the tabu list by now, and the current activity chain solution would not have passed the tabu check (step C).

If the check identifies an identical activity chain in the stored schedule solutions, it copy-pastes the stored schedule solution into the corresponding neighbourhood slot. Steps D2 to D6 can be dropped for this neighbourhood solution.

3.2.5 Selection of best neighbourhood schedule solution (step E)

Step E selects the best of the newly created, optimized and scored neighbourhood schedule solutions. The best schedule solution is the one with the highest utility:

- The schedule solution is added to the tabu list. Like this, later iterations’ solutions with the identical activity chain are deemed invalid and disregarded. This prevents PlanomatX from cycling.

- The schedule solution will be the base solution of the next outer loop iteration.

3.2.6 Finish of the algorithm (step F)

PlanomatX has two stop criteria (see section 3.2):

- The standard stop criterion is arriving at the externally given number of maximum outer loop iterations.

- The rather unusual stop criterion is fulfilled when no non-tabu solutions can be found during the neighbourhood creation (step B).

When either of the two criteria is fulfilled, step F is executed. Step F first selects the best schedule from those stored in the tabu list. Second, step F returns this schedule back to MATSim’s controller environment, replacing the schedule originally received.

\[17\] Remember that PlanomatX stores each outer loop iteration’s best schedule in the tabu list. Hence, selecting the best schedule of the tabu list means selecting the best schedule PlanomatX has found for the agent.
3.3 Test results of PlanomatX with Planomat

3.3.1 Fictive chessboard scenario

A simple, artificial chessboard scenario has been designed to test, verify and analyze PlanomatX’s functionality. The scenario has the following features (see also figure 3.14):

- **Network**: The network possesses a chessboard-like form. All links are unidirectional and have the same length (1km), the same free flow speed (27km/h), and the same capacity (10 cars/hour).

- **Facilities**: Three types of facilities exist:
  - 36 home and work facilities are located along the outer links. Home activities are possible around the clock. Work facilities are open from 8 am to 8 pm.
  - Five shopping facilities are distributed over the horizontal middle links. Shops are open from 10 am to 6 pm.
  - Five leisure facilities are located on the remaining horizontal middle links, between the shopping activities. Leisure facilities are open from 4 pm to 11 pm.

- **Agents’ schedules**: 324 agents “live” in the scenario. Their initial demand schedules resemble the following:
  - Half of the agents pursue an initial activity chain of “home-work-shopping-home”. The other agents pursue an initial activity chain of “home-work-leisure-home”. The activity timings are “home” from 0 am to 8 am, “work” from 8 am to 4 pm, “shopping”/”leisure” from 4 pm to 8 pm, and “home” from 8 pm to 12 pm.
  - Home and work locations are assigned so that nine agents are based at each home facility. Their work locations are distributed throughout the nine opposite work facilities.
  - Shopping and leisure locations are randomly allocated.

- **Agents’ knowledge**: Agents’ knowledge contains all four activity types. Obviously, locations of the three activity types in the initial schedule match locations in the schedules. The fourth location (either shopping or leisure) is assigned randomly. Assigned home and work activities are primary; shopping and leisure activities are secondary. As an example, figure 3.15 displays knowledge and initial schedule of agent 1.

---

18 Given the artificial nature of the scenario, shopping facilities are open, although no agent performs a work activity there.
19 Given the artificial nature of the scenario, leisure facilities are open, although no agent performs a work activity there.
20 36 home locations with nine agents each equal 324 agents in total.
Figure 3.14: Illustration of the fictive chessboard test scenario

Figure 3.15: Example of knowledge and initial schedule of agent 1 in the fictive chessboard scenario
Table 3.1: Parameter settings of utility function for all tests on the chessboard scenario

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Activity type</th>
<th>Home</th>
<th>Work</th>
<th>Education</th>
<th>Leisure</th>
<th>Shopping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Umax</td>
<td></td>
<td>60.0</td>
<td>55.0</td>
<td>40.0</td>
<td>35.0</td>
<td>12.0</td>
</tr>
<tr>
<td>Umin</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>6.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
</tr>
<tr>
<td>β</td>
<td></td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>β travelTime_car</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6.0</td>
</tr>
<tr>
<td>β travelTime_pt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6.0</td>
</tr>
<tr>
<td>β travelTime_walk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6.0</td>
</tr>
</tbody>
</table>

- **Utility function**: All tests on the chessboard scenario employ a preliminary utility function displayed in figure 3.16. Table 3.1 shows the corresponding parameters. Function parameters have been set in an effort to match the characteristics of MATSim’s existing utility function (see section 2.3.1.2).

### 3.3.2 Test I: Test of PlanomatX on chessboard scenario

A first test on the chessboard scenario is run to verify PlanomatX’s ability to optimize agents’ schedules. The following settings are applied:

- **MATSim settings**:
  - **MATSim iterations**: The test is run with 100 MATSim iterations.
  - **Replanning share**: Share of agents to be replanned in each iteration is 10%. PlanomatX is the only replanning strategy. The remaining 90% of agents select their current best schedule to be executed without any replanning.
  - **Storage of schedules**: Agents are allowed to store up to five schedules (see section 2.3.1.2).
  - **Traffic simulation**: MobSim is used for the traffic flow simulation (see section 2.3.1.2).

- **PlanomatX settings**:

---

21 This preliminary utility function does not rely on empirically estimated parameters. Instead, the function has enabled testing of PlanomatX and schedule recycling before the new utility function had been empirically estimated.

22 Note that, in this section, we refrain from conducting a sensitivity analysis of the settings. Section 4.6 presents such an analysis pertaining to the PlanomatX final version.
Figure 3.16: Plot of utility functions for all tests on the chessboard scenario

(a) Utility function for the performance of activities

- **Number of iterations**: PlanomatX is run with 20 outer loop iterations.
- **Neighbourhood size**: Step B’s neighbourhood size is 10 solutions.
- **Shares in the neighbourhood creation**: The shares of the change-number, change-order, and change-type sub-algorithms are 60%/20%/20%. The change-number sub-algorithm splits into 50% insertion and 50% removal of an activity.
- **Location choice**: Step D2 is run with reduced location choice.
- **Optimization of activity timings and mode choice**: Step D4 applies MATSim’s existing Planomat strategy for the optimization of activity timings and mode choice.

(b) Utility function for the performance of travel

- **Location choice**: Step D2 is run with reduced location choice.
- **Optimization of activity timings and mode choice**: Step D4 applies MATSim’s existing Planomat strategy for the optimization of activity timings and mode choice.

PlanomatX’s optimization impact is evaluated against a base test with MATSim’s existing replanning strategies. Instead of a 10% replanning share of PlanomatX, the base test was run with
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

shares of 4% Planomat, 3% Location Choice, and 3% Route Choice\(^\text{(c.p.)}\). Hence, the base test involves all replanning dimensions of PlanomatX except the activity chain dimension.

### 3.3.2.1 Development of utility scores, trip travel distances and times

Figure 3.17, upper chart, shows the development of average utility for agents’ executed schedules. Above all, PlanomatX successfully optimizes the schedules. Average executed utility rises from 6.07 utility points (initial demand schedules at iteration 0) to 134.27 utility points. The base test attains a utility level of only 117.53 points. Measured against the initial utility score at iteration 0, PlanomatX’s flexible activity chain dimension leads to a utility increase of 15%. The base test not reaching PlanomatX’s utility level underpins the significance of the activity chain dimension in MATSim’s replanning process. PlanomatX’s better utility performance is the result of a higher number of activities undertaken by agents and lower average trip travel distance. Figure 3.17, middle and lower charts, displays the development of average executed trip travel distance and average executed trip travel time. While the average trip travel distance per agent is about 9.7 km at iteration 0, it falls to about 6.5 km at iteration 100. The reduction is driven by two factors:

- **Location choice**: Location choice strategy may find more nearby locations for agents’ secondary activities.

- **Changes in the activity chains**: Flexibility in the activity chains allows withdrawal of activities with unfavourable locations and their replacement with activities that the location choice strategy can find on-route.

Average executed trip travel time falls from 2:56 hours (≈ 10,600 sec) to 27 min (≈ 1,600 sec). One reason is the lower average travel distance. Another reason is the co-evolutionary learning process, meaning that agents level out the network traffic load by varying locations, routes, and activity timings.

### 3.3.2.2 Initial demand and optimized schedules

Table 3.2 analyzes PlanomatX’s impact on agents’ activity chains. The new average length of agents’ activity chains is 5.06. No initial demand activity chain “home-work-shopping-home” has been kept and only 32 of 161 agents have kept their activity chain “home-work-leisure-home”. Instead, agents have extended their activity chains. The most popular optimized activity chain is now “home-work-shopping-leisure-home” adopted by 205 agents. Every agent

\(^{23}\)Note that the replanning strategies are called separately, not as an integrated strategy as implemented in PlanomatX.
Figure 3.17: Test I: Development of average executed schedule utility, trip distance and trip travel time

still performs work, in line with the definition of work being a primary activity. Figure 3.18 drills down, as an example, to agent 1. The figure shows the initial demand schedule and the optimized schedules after base test and PlanomatX:

- In the base test, the score improves from -34.97 to 101.38 utility points. The activity chain, by definition, remains stable. The agent still travels by car. Congestion has been relieved and travel times have fallen considerably compared to the initial schedule at iteration 0. Routes have not changed (not shown in figure).
Table 3.2: Test I: Analysis of activity chains

<table>
<thead>
<tr>
<th>Base test</th>
<th>PlanomatX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of occurrences Relative</td>
<td>Number of occurrences Relative</td>
</tr>
<tr>
<td>163 50.3%</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>161 49.7%</td>
<td>32 9.9%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>205 63.3%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>38 11.7%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>26 8.0%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>7 2.2%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>6 1.9%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>3 0.9%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>2 0.6%</td>
</tr>
<tr>
<td>0 0.0%</td>
<td>2 0.6%</td>
</tr>
</tbody>
</table>

4.00 5.06 Average number of activities

Figure 3.18: Test I: Initial demand schedule and optimized schedules of agent 1 (for sake of simplification, not showing activity locations and travel routes)

- The PlanomatX test produces a utility of 137.70 points. The initial demand activity chain “home-work-shopping-home” has been replaced by the new activity chain “home-work-shopping-leisure-home”. Activity timings have been adapted; for instance, the agent leaves home at 7.04 am rather than 8.00 am. Moreover, the agent now uses public transport rather than his car. The agent’s location choice - home and work activity locations - have not changed, as they are primary (not shown in figure). However, the agent now shops at facility 38 (instead of facility 37) and performs newly introduced leisure activity at facility 43 (not shown in figure).
3.3.2.3 Runtimes

The overall runtime was 5 min 17 sec for the base test and 3 hours 39 min for PlanomatX (factor 41). Thus, PlanomatX leads to a very substantial increase in runtime. Figure 3.19 shows runtime broken down by MATSim iteration for the two tests. The base test consumes about 0.7 sec per iteration for location choice, route choice and Planomat. MobSim and controller environment take about 2.4 sec. For the PlanomatX test, replanning takes almost 130 sec. The bulk of simulation runtime is due to Planomat, calling about 110 times per agent by PlanomatX (see below). MobSim and the controller environment remain stable. For each agent, steps D2 to D6 were run 110 times on average. This equals a 45% retrieval rate of previously optimized schedule solutions, since 200 neighbourhood schedule solutions were created per agent. The high retrieval rate cannot avoid Planomat running for more than 127 sec (96% of overall runtime), making Planomat the most significant runtime contributor. Nevertheless, its runtime contribution is traceable: each Planomat call takes about 0.04 sec. Multiplying 32 agents per iteration, with 110 Planomat calls per agent and 0.04 sec per call, results in an overall Planomat runtime of 127 sec per iteration.

---

24 20 iterations à 10 neighbourhood solutions, ignoring the rarely fulfilled stop criterion of finding no non-tabu solutions.

25 10% of 324 agents in the scenario.
3.3.2.4 Summary

PlanomatX achieves a higher utility score (+14%) than the base test because the activity chain dimension is an additional degree of freedom in the optimization. The higher utility score is a great result but the downside is PlanomatX’s very high runtime. Even for small scenarios like the chessboard scenario, simulation takes several hours.

3.4 Discussion

PlanomatX extends MATSim’s replanning functionality. It is a best response strategy that optimizes agents’ schedules comprehensively, i.e. the optimization refers to activity chains’ structure, location, route, and mode choices and activity timings. PlanomatX relies on the Tabu Search concept, but also incorporates existing MATSim replanning strategies such as location choice strategy, route choice strategy and Planomat Strategy.

A first test on the fictive chessboard scenario demonstrated PlanomatX’s optimization performance. In comparison with the the joint application of MATSim’s existing replanning strategies, PlanomatX reaches higher utility scores. This is possible since the structure of the activity chains (number, type and sequence of activities) is now a dimension of the co-evolutionary learning process. However, in its current form, PlanomatX requires disproportional runtimes, in spite of first runtime optimization (see e.g., efficient storage and retrieval of optimized schedules; section 3.2.4.5). Therefore, the following two chapters will deal only with methods to squeeze PlanomatX’s runtime, making PlanomatX feasible for large-scale scenarios:

- Chapter 4 will emphasise runtime reduction with regard to mode choice and activity timing (step D4). It will again present a Tabu Search heuristic to replace the current used Planomat strategy.
- Chapter 5 introduces a schedule recycling concept that avoids running PlanomatX for each agent individually.
Chapter 4

Optimization of mode choices and activity timings: TimeModeChoicer

TimeModeChoicer is a strategy to optimize a schedule’s mode choices and activity timings. It can be used both as a PlanomatX sub-algorithm (PlanomatX’s step D4, see section 3.2.4.3) and as a stand-alone replanning strategy.

4.1 Problem formulation

The objective function of the TimeModeChoicer is identical to the PlanomatX algorithm (see section 3.1):

$$\max U_{\text{total}} = \max \left[ \sum_{i=1}^{n} U_{\text{act},i} + \sum_{i=1}^{n} U_{\text{late},i} + \sum_{i=1}^{n} U_{\text{travel},i} \right]$$

(4.1)

where $U_{\text{total}}$ is total utility of the given schedule; $n$ is the number of activities/trips; $U_{\text{act},i}$ is the (positive) utility gained from performing activity $i$; $U_{\text{late},i}$ is the (negative) utility gained from arriving late at activity $i$; and $U_{\text{travel},i}$ is the (negative) utility gained from travelling trip $i$.

In comparison with the PlanomatX algorithm, the TimeModeChoicer has less freedom to maximize utility. It deals only with optimizing modes chosen and activity timings. To calculate best mode choices, it also (partly) integrates route choice dimension. The resulting problem is, again, a mixed-integer, non-linear, and non-convex problem requiring a heuristic solution approach.
To simplify, the description of TimeModeChoicer’s heuristic solution approach is explained in two steps:

1. **Optimization of activity timings**: Forming the basic structure of the overall TimeModeChoicer, optimization of activity timings will be outlined in section 4.2.

2. **Integration of mode choice**: Section 4.3 will present how mode choice dimension is integrated, completing the TimeModeChoicer.

### 4.2 Heuristic solution algorithm - Optimization of activity timings

Like the PlanomatX algorithm, TimeModeChoicer relies on the Tabu Search approach. When choosing a solution approach, the same logic applies for TimeModeChoicer as for PlanomatX (see section 3.2). Figure 4.1 illustrates the high-level process flow of the TimeModeChoicer vis-à-vis optimization of activity timings. One can observe three major changes in comparison with the PlanomatX process flow:

- The inner loop no longer exists. This is reasonable, since neighbourhood solutions require no optimization of lower tier dimension.

- The tabu check has been dropped. Instead, elaborate rules in neighbourhood creation ensure that cycling cannot happen. This reduces the algorithm’s runtime.

- A new stop criterion has been implemented, verifying whether or not the score has improved over a number of n’ last iterations. If not, the algorithm stops. Again, this reduces the algorithm’s runtime.

#### 4.2.1 Initialization of the algorithm (step A)

TimeModeChoicer receives the schedules to be optimized either from the PlanomatX algorithm or from MATSim’s controller environment. Each schedule is part of a larger agent-specific data set also including information on the agent’s knowledge and socio-economics (see previous chapter, figure 3.3). TimeModeChoicer makes no a priori assumptions about the accuracy of a schedule. Instead, it checks activity timings of the received schedule to ensure they are sound:

---

1 We will see in section 4.3.1 that the TimeModeChoicer treats mode choice dimension as a lower tier decision of the activity timings dimension. Hence, optimization of mode choices is somewhat similar to an inner loop. However, it is implemented differently.
Figure 4.1: TimeModeChoicer process flow - optimization of activity timings

- Activities and/or trip timings must not overlap.
- Activity durations must be equal to or above the minimum durations for each activity type.

If the TimeModeChoicer identifies activities/trips breaking the rules, it adjusts them and sets the resulting schedule as base solution for the first iteration. If no feasible solution can be found, the TimeModeChoicer returns the schedule without further modification.

4.2.2 Neighbourhood creation (step B)

TimeModeChoicer’s neighbourhood creation deals with generating a list $N$ of $K$ neighbourhood schedule solutions with different activity timings. Different activity timings are achieved by increasing/decreasing durations of two base schedule activities. Depending on the current iteration’s number $n$, the TimeModeChoicer executes a full neighbourhood enumeration (step B1), or a directed neighbourhood search (step B2). Steps B1 and B2 steer the time variations and call the operators Bx.1 to Bx.3 to execute them (see figure 4.2).

4.2.2.1 First iteration ($n=1$): Full neighbourhood enumeration (step B1)

In the first iteration, TimeModeChoicer enumerates all $2^K$ possible combinations of increasing/decreasing an activities pair’s durations (see figure 4.3). Full enumeration helps deter-

---

2Only single combinations are considered, no multiple combinations at one time.
mine initial direction of the steepest ascent. For each pair of activities, step B1 calls both steps Bx.1 (increase-time) and Bx.2 (decrease-time). The size of iteration’s 1 neighbourhood is thus $(Z - 1) \times (Z - 2) \times \ldots \times 1$, where $Z$ is the number of activities in the base activity chain.

### 4.2.2.2 Second iteration onwards (n>1): Directed neighbourhood search (step B2)

From the second iteration on, the TimeModeChoicer directs its neighbourhood search. The size of the neighbourhood must be given externally. In the base solution (= last iteration’s selected best solution), $z_{inc}$ denotes the activity whose duration was increased, and $z_{dec}$ the activity whose duration was decreased. The following rules apply:

- The first third of the neighbourhood is filled with solutions for which the duration of activity $z_{inc}$ is again increased. Half of that third is searched by calling the decrease-time operator for lower-positioned activities; the lower-positioned activities are shortened in time, activity $z_{inc}$ is lengthened. The other half is searched by calling the increase-time operator for activity $z_{inc}$; activity $z_{inc}$ is increased, subsequent activities are shortened.

- The second third of the neighbourhood is filled with solutions for which the duration of activity $z_{dec}$ is again decreased. The search mechanism is the same as for the first third but with opposite signs.

- Finally, the last third of the neighbourhood is filled with solutions for which the duration of other activities is increased or decreased. This ensures that some flexibility is maintained in the direction the algorithm proceeds.
Figure 4.3: Steps B1 and B2: Illustration of the neighbourhood moves of the TimeMode-Choicer

Rules above and some minor (not specifically described) if-clauses ensure that direct turn-back moves are impossible and that the probability of cycling is minimized. As a consequence, the tabu check can be dropped, creating a positive effect on the algorithm’s runtime performance.

### 4.2.2.3 Time variation operators (step Bx)

Both steps B1 and B2 use increase-time and decrease-time operators to realize activity timing variations. The two operators are complemented by the swap-times operator, a fall-back if a
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling 2010

normal increase/decrease move is not possible.

**Increase-time operator (step Bx.1)**

The increase-time operator increases the duration of an activity \( z \) by an offset time, generally 30 minutes (see section 4.5.2.3). Balancing overall duration of the schedule, it decreases duration of another activity \( z' \) at a higher activity chain position accordingly, i.e. \( z < z' \). This implies the following sub-steps:

1. The operator checks whether activity \( z' \) would fall below its minimum time if reducing its current duration by the offset time. If so, the operator stops and calls the swap-times operator (step Bx.3). Otherwise, it proceeds.

2. The operator increases duration of the activity \( z \) by the offset time. The new end time of activity \( z \) is equal to the new departure time of the following trip. Using the departure time, the operator re-calculates the new trip travel time from activity \( z \) to activity \( z + 1 \). Travel time may differ from the original, due to different traffic loads. Using the new arrival time, it adjusts start and end times of activity \( z + 1 \) where the new end time is the maximum function of the arrival time plus its minimum time, or the former end time plus the offset. The operator then re-calculates the travel time from activity \( z + 1 \) to activity \( z + 2 \) at the new departure time, and so forth.

3. Reaching activity \( z' \), the operator ensures that activity \( z' \) stays above its minimum time. If this is not the case (e.g., because traffic at the modified travel departure times is much denser), it also shortens the following activity \( z' + 1 \) until that one also reaches its minimum time. If this is not sufficient, the operator eventually declines the operation and sends a corresponding message back to steps B1/B2.

**Decrease-time operator (step Bx.2)**

The decrease-time operator decreases the duration of an activity \( z \) and, balancing the overall duration of the schedule, increases the duration of another activity \( z' \) at a higher activity chain position accordingly, i.e. \( z < z' \). The decrease-time operator works identically to the increase-time operator, but with opposite signs.

**Swap-times operator (step Bx.3)**

If the first sub-step of the increase/decrease-time operators (see above) indicates that the activity to be shortened would fall below its minimum time, the increase/decrease-time operators call the swap-times operator (see figure 4.4). This operator just swaps the durations of the two activities. However, the swap-times operator must take care to not decrease the activity that
is now to be shortened below its minimum time. If this should happen, it swaps the durations minus that portion putting the activity below minimum time.

The swap-times operator ensures that no dead ends exist when moving through the activity timings landscape. It assumes an important role by not letting the TimeModeChoicer get stuck in local optima.

### 4.2.2.4 Scoring of the neighbourhood schedule solutions

The newly derived neighbourhood schedule solutions need to be scored, i.e. their utilities determined. Obviously, the same utility function must be used as in section 3.2.4.4.

### 4.2.3 Selection of best neighbourhood schedule solution (step C)

After all neighbourhood schedule solutions have been created and scored, step C selects the best schedule solution featuring the highest utility. This schedule solution is set as next iteration’s base solution\(^3\).

However, prior to proceeding to the next iteration, step C checks the incremental improvement

\(^3\)In the neighbourhood creation (step B), the TimeModeChoicer writes only modified trip timings to the created schedule solutions, not activity timings. This is sufficient, since the scoring module draws the activity durations solely from trip departure and arrival times. Therefore, activities of the selected best schedule solution must still be updated with modified timings prior to setting the solution as the next iteration’s base solution. Since only the selected best schedule solution must be updated with the modified activity timings (not all created neighbourhood schedule solutions), this procedure safes runtime.
over the last $n'$ iterations. If it is above an externally given threshold, the TimeModeChooser continues iterating. Otherwise, the TimeModeChooser breaks the iterations and proceeds directly to step D.

4.2.4 Finish of the algorithm (step D)

The TimeModeChooser has two stopping criteria (see section 3.2):

- The standard stopping criterion is arrival at the externally given maximum number of iterations.
- The second stopping criterion is the threshold criterion just described in step C.

When either of the two criteria is met, step D first selects the best schedule of all iterations. Second, step D returns this schedule to the PlanomatX algorithm, or to MATSim’s controller environment, replacing the schedule originally provided.

4.3 Heuristic solution algorithm - Integration of mode choice

4.3.1 Impact of mode choice dimension on solution space

The TimeModeChooser implements a sub-tour based mode choice. When activity timings of a schedule solution have been varied, modes of the schedule solution’s sub-tours may be verified and optimized. TimeModeChooser offers two options.

4.3.1.1 Full enumeration of mode choice combinations

Full enumeration tests all possible combinations of sub-tour modes of a schedule solution for their utilities and selects the combination with highest utility. TimeModeChooser’s solution space increases by a factor $m^s$, where $m$ is the number of available modes and $s$ the number of sub-tours in the schedule solution.

4.3.1.2 Mode choice for affected subtour(s) only

When TimeModeChooser’s neighbourhood creation increases and decreases activity durations, it moves some trips forward or backward in time. The moved trips are those positioned between
varied activities. Optimization of mode choices should focus on these trips. Mode choice for affected subtour(s) puts its emphasis on “outer” trips of those moved (see figure 4.5). It optimizes mode choice only for the sub-tours belonging to outer trips, to be labelled “affected sub-tours”. The following combinations of affected sub-tours are possible:

- One sub-tour is affected when only one trip is moved (the two varied activities are positioned next to each other) or when the two outer loops belong to the same sub-tour.
- Two sub-tours are affected when at least two trips are moved and the two outer trips belong to different sub-tours.

The factor that increases TimeModeChoicer’s solution space is \( m^{s_{\text{affected}}} \), where \( m \) is the number of available modes and \( s_{\text{affected}} \) the number of affected sub-tours in the schedule solution (\( s_{\text{affected}} < s \)).

### 4.3.2 TimeModeChoicer’s four alternatives to integrate mode choice

TimeModeChoicer offers four alternatives to integrate these mode choice options into its process flow (see figure 4.6). The alternative to be used must be given externally. The choice of alternative is a trade-off between desired accuracy of results and runtime.

#### 4.3.2.1 MCs: Standard alternative

The standard alternative optimizes mode choice only for affected sub-tours of the schedule solution selected as best neighbourhood solution in step C, and it is the simplest mode choice alternative TimeModeChoicer offers.

#### 4.3.2.2 MC1: Extension alternative 1

Extension alternative 1 also considers only the schedule solution selected as best neighbourhood solution in step C, but optimizes mode choice for all sub-tours, not just for affected sub-tours.

#### 4.3.2.3 MC2: Extension alternative 2

Extension alternative 2 activates every time the increase-time operator is called (step Bx.1). Thus, it applies not only to the schedule selected as best neighbourhood solution in step C. It
Figure 4.5: Illustration of mode choice combinations for affected sub-tour(s) only (assuming three available modes: car, PT, walk)

Applies instead to about a third of created neighbourhood solutions. Extension alternative 2 optimizes mode choice only for affected sub-tours.

4.3.2.4 MC3: Extension alternative 3

Extension alternative 3 is the most comprehensive alternative. It optimizes mode choice for affected sub-tours of all neighbourhood solutions, regardless of whether they were created by increase-time, decrease-time, or swap-times operator.

---

\[\text{Number of applications is reduced by the number of swap-times operator calls (step Bx.3).}\]
4.3.3 A-priori assumptions for walking

Reducing the mode choice’s solution space, the TimeModeChoicer offers the possibility to make a-priori assumptions about the mode walk:

- One may set a maximum distance for all walk sub-tours, for instance 2 km. For all sub-tours featuring longer distances, walk mode is skipped when checking for the best mode.

- All sub-tours with a distance of 0 meter are automatically assigned to walk mode. Sub-tours with a distance of 0 meter occur often, e.g., when two consecutive activities take place at the same location (i.e. same facility or two facilities on the same link). From 1 meter sub-tour distance, all modes are evaluated so as not to artificially limit the solution space.
A-priori assumptions on walk mode save substantial runtime. They remove those mode combinations’ utility checks containing sub-tours where one of the two rules above hold true.

### 4.4 Aligned interface between TimeModeChoicer and PlanomatX

#### 4.4.1 Feasibility of schedules

The TimeModeChoicer’s step A verifies feasibility of the schedule received. It adjusts schedule activity timings if they break non-overlapping or minimum-duration rules (see section 4.2.1). If feasibility of the received schedule cannot be achieved solely through correction of activity timings, the TimeModeChoicer also tries to rectify the received schedule through mode choice optimization. Step A calls the full mode choice enumeration (see section 4.3.1.1). In the event mode optimization is unsuccessful, the TimeModeChoicer declines the received schedule, marks it as “not feasible”, and returns it to the PlanomatX/controller environment (without changing the original). PlanomatX reacts as follows:

- **PlanomatX’s step A**: When the TimeModeChoicer was called by PlanomatX’s step A, PlanomatX shortens the schedule’s activity chain. It removes the last activity (subject to the removability of primary activities; see section 3.2.2.1) and again sends the schedule to the TimeModeChoicer. In case the TimeModeChoicer still declines the schedule, PlanomatX removes another activity, and so forth. The procedure ensures that PlanomatX always generates a feasible base solution for its outer loop iteration 1.

- **PlanomatX’s step D4**: When the TimeModeChoicer is called by PlanomatX’s step D4, PlanomatX does not shorten the activity chain of the neighbourhood schedule solution. It just adopts the “not feasible” mark. The later tabu check (step C) then drops the solution from the neighbourhood, similar to the solutions marked “not new”.

#### 4.4.2 Employment of router strategy

PlanomatX’s step D3 (call of the router strategy) is executed only when step D4 calls the Planomat strategy. It is omitted when step D4 calls the TimeModeChoicer, because TimeModeChoicer conducts mode choice anyway. This is a leaner algorithm structure that saves some runtime.
4.4.3 Employment of scoring module

PlanomatX’s step D5 is also omitted when step D4 calls the TimeModeChoicer. This is reasonable, since the TimeModeChoicer’s step B3 scores the schedule solutions anyway. In particular, it has scored the final best solution handed back to the PlanomatX algorithm. The utility score is also simply handed back, making PlanomatX’s step D5 redundant and saving runtime.

4.5 Test results of the TimeModeChoicer as a stand-alone replanning strategy

The TimeModeChoicer replicates Planomat’s functionality, but aims for faster runtimes. Two tests were conducted to analyze how TimeModeChoicer performance versus Planomat:

- Given some base settings, test II evaluates performance of the TimeModeChoicer on the chessboard scenario. A Planomat test serves as benchmark.
- Test III focuses on the TimeModeChoicer and analyzes its behaviour with different parameter settings. Test III is also conducted on the chessboard scenario.

4.5.1 Test II: Test of TimeModeChoicer on chessboard scenario

Test II analyzes performance of the TimeModeChoicer on the chessboard scenario and compares it with the Planomat strategy. The following settings are applied:

- MATSim settings:
  - MATSim iterations: The test is run with 100 MATSim iterations.
  - Replanning share: The share of agents to be replanned in each iteration is 10%. TimeModeChoicer is the only replanning strategy. The remaining 90% of agents select their current best schedule to be executed without any replanning.
  - Storage of schedules: Agents are allowed to store up to five schedules (see section 2.3.1.2).
  - Traffic simulation: MobSim is used as traffic simulation (see section 2.3.1.2).

- TimeModeChoicer settings:
  - Number of iterations: TimeModeChoicer is run with 30 iterations. n’ is 5 iterations (i.e. the algorithm stops if the score has not improved over the 5 last iterations).
– **Neighbourhood size**: Step B’s neighbourhood size is 10 slots.
– **Offset time**: The offset time is 0.5 hours (1800 seconds).
– **Minimum activity durations**: Minimum activity durations are 2 hours for home and 1 hour for work, shopping, and leisure.
– **Mode choice**: Available modes are car, public transport, and walking. Standard mode choice (MCs) is applied. Maximum walking distance is 2 km.

• Planomat settings (identical with test I):
  – **Number of iterations**: Planomat is run with 50 GA iterations.
  – **Mode choice**: Available modes are car, public transport, and walking (identical to TimeModeChoicer).

### 4.5.1.1 Development of utility scores, trip travel distances and times

Figure [4.7](#) contrasts TimeModeChoicer’s optimization curve with Planomat’s optimization curve. Both TimeModeChoicer and Planomat achieve roughly the same utility level (TimeModeChoicer: 119.06 points; Planomat: 118.56 points). Planomat’s curve rises faster at the beginning of the optimization. TimeModeChoicer’s curve catches up around iteration 15. The two lower charts of figure [4.7](#) illustrate development of average executed trip travel distances and times. The two distance curves virtually overlap, as do the two travel time curves. Planomat’s version has small advantages; its travel time curve falls slightly below TimeModeChoicer’s during the first 50 or so iterations. The generally strong decrease of travel times is due to a wider spread of travel departure times (see e.g., early work start of shopping agents versus late work start of leisure agents) and the use of public transport as an alternative mode to car.

### 4.5.1.2 Initial demand and optimized schedules

Figure [4.8](#) demonstrates the impact of optimization on average activity timings. One clearly observes that overall time spent in activity performance has substantially increased, while trip travel times have fallen. TimeModeChoicer’s optimized schedules resemble the following:

• **Shopping activity chains**: For schedules with a shopping activity, agents now leave home at about 7:20 am. Work activities have an average duration of 6.5 hours and end at about 3 pm. Shopping activities take 2.5 hours on average, ending before shops close at 6 pm. Since agents try to reach shops on time, they start working earlier. The shortened first home activity is balanced by the longer second home activity.
Figure 4.7: Test II: Development of average executed schedule utility, trip distance and trip travel time

- **Leisure activity chains:** Agents with a leisure activity sleep in, leaving home only when shopping agents have already reached their work facilities. Leisure activities take place from about 6 pm till 10 pm, probably linked to work closure time (6 pm). Leisure closure time (11 pm) seems to be irrelevant.

Mode choice is split about equally between car and public transport, though differences exist according to activity chain; agents with a shopping activity tend to use the car. Agents with a
leisure activity prefer public transport. Walk mode is never chosen, since all sub-tours involve distances over 2 km.

Optimized Planomat schedules look very similar. The first home activity is marginally shorter. Work activity (and leisure activity) have been slightly lengthened. Mode choices favour the car slightly less for shopping schedules, but are almost identical with TimeModeChoicer choices for leisure schedules.

As an example, figure 4.9 focuses on agent 1. The figure shows initial demand schedule at iteration 0, as well as the two optimized schedules. The TimeModeChoicer schedule features a higher utility score (112.15 utility points) than the Planomat schedule (107.22 utility points). Key aspects of the schedules are:

- **TimeModeChoicer schedule**: The agent leaves home at 7:30 am. Work activity has been shortened to 7:02 hours. Shopping activity has been shortened to 2:00 hours. The second home activity has been lengthened to 5:41 hours. In addition, the agent has changed from car to public transport.

- **Planomat schedule**: The agent leaves home later at 8:54 am, spending 7:13 hours at work, and only 1:15 hours on shopping. Agent returns home at 19:09 pm, about 50 minutes later than in the TimeModeChoicer schedule. Mode and route choices have not changed.

In both schedules, activity chains and locations (not shown in figure) are unchanged.
Figure 4.9: Test II: Initial demand schedule and optimized schedules of agent 1 (for sake of simplification, not showing activity locations and travel routes)

Figure 4.10: Test II: Comparison of runtime shares of TimeModeChoicer and Planomat

4.5.1.3 Runtime

The overall runtime of the TimeModeChoicer test was 4:22 minutes. Each MATSim iteration took about 2.6 seconds. TimeModeChoicer accounts for 0.2 sec of an iteration’s runtime (see figure 4.10); roughly 6 msec per agent. The runtime of the Planomat test was 4:50 min. Each MATSim iteration took about 3.5 sec. Planomat accounts for 1.1 sec, roughly 34 msec per agent. TimeModeChoicer is thus about six times faster than Planomat.
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling 2010

Figure 4.11: Test III: Impact of variation of TimeModeChoicer’s neighbourhood size on development of utility scores and runtimes

<table>
<thead>
<tr>
<th>Neighbourhood size</th>
<th>Utility after 100 iterations (ms)</th>
<th>Relative change against neighbourhood size of 10 slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>118.53, 4.3</td>
<td>-31.04%</td>
</tr>
<tr>
<td>5</td>
<td>119.03, 5.6</td>
<td>-10.90%</td>
</tr>
<tr>
<td>10</td>
<td>119.06, 6.3</td>
<td>0%</td>
</tr>
<tr>
<td>12</td>
<td>119.02, 6.7</td>
<td>0.02%</td>
</tr>
</tbody>
</table>

4.5.1.4 Summary

TimeModeChoicer and Planomat perform equally well on achieved utility level, structure of optimized schedules, and trip distances and travel times. They differ in runtime. TimeModeChoicer is about six times faster than Planomat.

4.5.2 Test III: Variation of TimeModeChoicer settings on chessboard scenario

Test III focuses on the TimeModeChoicer and examines its behaviour under different parameter settings. In particular, it examines TimeModeChoicer’s performance when size of its neighbourhood, number of iterations (including the stopping criterion), offset time, or type of mode choice are varied.

4.5.2.1 Variation of neighbourhood size

Figure 4.11 illustrates development of utility and runtime depending on the size of TimeModeChoicer’s neighbourhood. One can observe that the neighbourhood size has hardly any effect on utility score after 100 iterations. Yet, the runtime per TimeModeChoicer call changes; it varies between 4.3 msec and 6.7 msec. This is a range of -31.0% to +7.2% versus the base case of 10 neighbourhood solutions with a runtime of 6.3 msec.

5The length of agents’ initial demand activity chains explains why no larger neighbourhoods were tested; the activity chain features only 5 activities. A full enumeration of increasing/decreasing durations of chain’s activities comprises 12 combinations. Any neighbourhood of 13 slots or more would be oversized.
4.5.2.2 Variation of number of iterations and stop criterion

Figure 4.12 illustrates the development of utility scores and runtimes depending on the number of iterations and the stopping criterion. Again, neither parameter has any significant effect on utility level after 100 iterations. Runtime per TimeModeChoicer call varies between 2.7 msec and 12.1 msec, a range of -56.5% to +90.9% versus the base case.

4.5.2.3 Variation of offset time

Figure 4.13 illustrates development of utility and runtime depending on TimeModeChoicer’s offset time. Utility after 100 iterations becomes maximum using an offset time of 30 min. Run-times vary between 7.8 msec (15 min offset time) and 5.6 msec (45 min offset time). The range of runtimes is smaller than in previous variation tests because offset time has no direct impact on any runtime-relevant elements of the TimeModeChoicer, as opposed to, for instance, number of iterations or neighbourhood size. Indirect effects may just be an earlier/later fulfilment of the stopping criterion.
Figure 4.13: Test III: Impact of variation of TimeModeChoicer’s offset time on development of utility scores and runtimes

<table>
<thead>
<tr>
<th>Offset time (minutes)</th>
<th>Average executed score after 100 iterations</th>
<th>TimeModeChoicer runtime per agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>118.41</td>
<td>7.8</td>
</tr>
<tr>
<td>30</td>
<td>119.06</td>
<td>6.3</td>
</tr>
<tr>
<td>45</td>
<td>118.62</td>
<td>5.6</td>
</tr>
<tr>
<td>60</td>
<td>118.93</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Relative change (in %) against offset time of 30 minutes

- Offset time of 15 minutes: -0.55
- Offset time of 30 minutes: 23.34
- Offset time of 45 minutes: -11.06
- Offset time of 60 minutes: -0.11

Figure 4.14: Test III: Impact of variation of mode choice on development of utility scores and runtimes

<table>
<thead>
<tr>
<th>Type of mode choice</th>
<th>Average executed score after 100 iterations</th>
<th>TimeModeChoicer runtime per agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCs</td>
<td>119.06</td>
<td>6.3</td>
</tr>
<tr>
<td>MC1</td>
<td>119.06</td>
<td>15.0</td>
</tr>
<tr>
<td>MC2</td>
<td>119.22</td>
<td>27.9</td>
</tr>
<tr>
<td>MC3</td>
<td>119.44</td>
<td>341.93</td>
</tr>
</tbody>
</table>

Relative change (in %) against standard mode choice (MCs)

- Standard mode choice (MCs): 0%
- Extension 1 (MC1): 136.97%
- Extension 2 (MC2): 0.14%
- Extension 3 (MC3): 0.32%

4.5.2.4 Variation of mode choice

Figure 4.14 illustrates development of utility and runtime depending on type of mode choice. Standard mode choice (MCs) and extension 1 (MC1) must, by definition, yield the same results since all schedules have only one sub-tour. Extended mode choices 2 and 3 are the only settings of all four tests that produce “perceptibly” better utility scores (+0.14%/+0.32%). However, they also account for the highest runtime increases (+137%+/342%).
4.5.2.5 Summary

Very few score improvements are possible when varying TimeModeChoicers parameter settings. Only extended mode choice settings produce slightly better scores than the base case. However, a key result of test IV is that TimeModeChoicer’s number of iterations and neighbourhood size may be reduced without major score losses. The reduction is positive, as it produces lower runtimes.

4.6 Test results of PlanomatX with TimeModeChoicer

TimeModeChoicer’s core role is to serve as a runtime-efficient Planomat replacement for PlanomatX’s step D4. We have conducted two tests on the chessboard scenario to analyze PlanomatX performance when employing the TimeModeChoicer:

- Test IV features “PlanomatX with TimeModeChoicer” (PlX/TMC) and compares it with the earlier alternative “PlanomatX with Planomat” (PlX/Pl) from test I.
- Test V focuses solely on the PI/TMC alternative and analyzes its behaviour with different parameter settings.
- Test VI repeats test IV but is a “real-world” simulation run on the greater Zurich scenario. It also introduces MATSim’s new utility function from chapter 6.

4.6.1 Test IV: Test of PlanomatX with TimeModeChoicer on chessboard scenario

Test IV evaluates alternative “PlanomatX with TimeModeChoicer” (PlX/TMC) and compares it with the former alternative “PlanomatX with Planomat” (PlX/Pl) from test I. All MATSim, PlanomatX, and TimeModeChoicer settings of the PlX/TMC alternative are those from test I, except for PlanomatX’s step D4 drawing on the TimeModeChoicer.

4.6.1.1 Development of utility scores, trip travel distances and times

Figure 4.15 shows the utility development of PI/TMC and PI/Pl alternatives. Initially, utility of the PI/Pl alternative rises faster than the PI/TMC one. Later, PI/Pl and PI/TMC alternatives reach about the same utility (135 points). The difference in the two trip distance curves is substantial (see figure 4.15). The PI/TMC alternative produces higher average trip
distances than the PIX/Pl alternative. However, the higher distance does not translate into longer trip travel times. The average trip travel time curves almost overlap.

### 4.6.1.2 Initial demand and optimized schedules

Table 4.1 contrasts optimized activity chains of the two alternatives. Average length of the PIX/TMC activity chains is 5.30 versus 5.06 for the the PIX/Pl alternative. TimeModeChoicer guides PlanomatX toward more, but shorter activities, leading to a slightly higher utility score (see previous section). Further, the PIX/TMC alternative results in a significantly higher entropy among activity chains, leading to 17 different chains, compared to only 10 chains for the PIX/Pl alternative. Figure 4.16 once again emphasizes agent 1. It shows its initial demand schedule at
Table 4.1: Test IV: Analysis activity chains

<table>
<thead>
<tr>
<th>Base test</th>
<th>PIX/PI</th>
<th>PIX/TMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of occurrences</td>
<td>Number of occurrences</td>
<td>Number of occurrences</td>
</tr>
<tr>
<td>Relative</td>
<td>Relative</td>
<td>Relative</td>
</tr>
<tr>
<td>163</td>
<td>50.3%</td>
<td>0</td>
</tr>
<tr>
<td>161</td>
<td>49.7%</td>
<td>32</td>
</tr>
<tr>
<td>16</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>38</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>26</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
</tbody>
</table>

Average number of activities: 4.00, 5.06, 5.30

Figure 4.16: Test IV: Initial demand schedule and optimized schedules of agent 1 (for sake of simplification, not showing activity locations and travel routes)

- In the PIX/Pl alternative, the score improved from -34.97 to 137.70 utility points. Agent 1 follows a “home-work-shopping-leisure-home” activity chain, using public transport during the day.
The PIX/TMC schedule yields 141.71 utility points. Activity chain and mode choice are similar, but shopping and leisure take place at different facilities (not shown in figure). Somewhat shorter travel times cause the slightly higher utility score.

### 4.6.1.3 Runtime

Comparison of the PIX/TMC and PIX/Pl alternatives is most important for the runtime experience. While the overall PIX/Pl runtime has been 3:40 hours, it is only 42:20 min for the PIX/TMC alternative. The runtime per MATSim iteration has decreased from 132 sec to 25.4 sec (see figure 4.17), a decrease of 81%. The reduction is solely driven by the TimeModeChoicer. It reduces the PlanomatX runtime per agent from 4.05 sec to 712 msec.

### 4.6.1.4 Summary

PIX/TMC alternative exceeds PIX/Pl alternative both in utility achieved and runtime consumed. PIX/TMC achieves a slightly higher utility score and it runs about five times faster than PIX/Pl. This is a positive outcome, most beneficial for simulation of large-scale scenarios.
4.6.2 Test V: Variation of PlanomatX/TimeModeChoicer settings on chessboard scenario

Test V focuses on PIX/TMC alternative, examining different parameter settings:

- Sections 4.6.2.1 through 4.6.2.3 refers only to variation of PlanomatX settings.
- Section 4.6.2.4 deals with joint variation of PlanomatX and TimeModeChoicer settings.

All results are compared against the base PIX/TMC alternative of test IV.

4.6.2.1 Variation of PlanomatX neighbourhood size

Figure 4.18 illustrates development of utility scores and runtimes depending on size of the PlanomatX neighbourhood. Scores of alternative neighbourhood settings range from 132.04 utility points (5 neighbourhood slots) to 140.43 utility points (20 neighbourhood slots), a change of -2.8% to +3.4% compared to the base case with 10 neighbourhood slots. Runtime per agent varies from 332 ms (5 neighbourhood slots) to 1,311 ms (20 neighbourhood slots), a change of -53.3% to +84.2% compared to the base case.

---

6For runtime reasons, all runs of test V have been conducted with 20 MATSim iterations only. Utility figures have been scaled up according to base alternative utility difference between iteration 20 (127.25 points) and iteration 100 (135.78 points, +6.7%).
4.6.2.2 Variation of number of PlanomatX iterations

Figure 4.19 illustrates development of utility scores and runtimes depending on the number of PlanomatX iterations. Scores after 20 iterations range from 131.53 utility points (5 iterations) to 139.10 utility points (50 iterations), a change of -3.1% and +2.5% against the base case with 20 iterations. Runtime per agent varies from 176 ms (5 iterations) to 1,685 ms (50 iterations), a change of -75.3% to +136.6% against the base case with 10 neighbourhood slots.

4.6.2.3 Variation of location choice

Figure 4.20 illustrates development of utility scores and runtimes depending on location choice type. Full location choice produces a slightly lower score (134.52 utility points) than the reduced location choice (135.78 utility points). This is counter-intuitive, but due to the fact that location choice is a constraint-based strategy. Apparently, when the full schedule is handed over to the location choice strategy, more downside potential exists than when just the modified activities are handed over. In the latter case, the location choice module finds nearby locations, in the former case not necessarily. The iterating location choice (iteratingLC3) exceeds reduced location choice confirming the impact of PlanomatX’s step D6. Runtimes are nearly identical for reduced location choice (712 msec) and full location choice (711 msec), because share of location choice is insignificant in the overall PlanomatX runtime (compare with figure 4.17). Runtime of the iterating location choice (1,977 msec) is in line with the additional

---

7LC3 means that a set of three schedules is generated and tested (the base procedure of steps D2 to D5 plus another two iterations of PlanomatX’s step D6).
Figure 4.20: Test V: Impact of variation of location choice on utility score development and runtime

<table>
<thead>
<tr>
<th>Type of location choice</th>
<th>Average executed score after 20 iterations</th>
<th>PlX/TMC runtime per agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>reducedLC</td>
<td>135.78</td>
<td>712</td>
</tr>
<tr>
<td>fullLC</td>
<td>134.52</td>
<td>711</td>
</tr>
<tr>
<td>iteratingLC3</td>
<td>136.12</td>
<td>1,977</td>
</tr>
</tbody>
</table>

Relative change (in %) against reducedLC:
- Utility ms: iteratingLC3 = 0.25, fullLC = -0.92, reducedLC = 0
- PlX/TMC runtime per agent: iteratingLC3 = -0.14, fullLC = -0.92, reducedLC = -0.14

4.6.2.4 Joint variation of PlanomatX and TimeModeChoicer settings

This section analyzes the behaviour of the PlX/TMC alternative when PlanomatX and TimeModeChoicer settings are varied simultaneously. Using previous tests, the analysis emphasizes number of iterations (including TimeModeChoicer’s stop criterion) and algorithms’ neighbourhood size.

Runtime-increasing settings

Figure 4.21 illustrates test results when applying runtime-increasing settings. Beyond the base case, four settings were examined. Settings A and B underline that it is better to intensify PlanomatX iterations than TimeModeChoicer iterations. This holds true for both final score and runtime. Settings C and D show that the neighbourhood enlargement contributes to a significantly better score. Setting D, though, requires substantially more runtime.

Runtime-reducing settings

Figure 4.22 illustrates the results of applying runtime-reducing settings. Beyond the base case, five settings were examined. Setting E confirms that reduction of number of TimeModeChoicer iterations is a no-regret move. The runtime is cut by half, the final score remains stable. Settings C and D show that it is better to continue reducing the number of PlanomatX iterations.

---

8The slight increase against the base case is due only to the random generator.
Figure 4.21: Test V: Impact of joint application of runtime-increasing PlanomatX and Time-ModeChoicer settings

<table>
<thead>
<tr>
<th>Settings</th>
<th>PIX Iter.</th>
<th>PIX Neigh.</th>
<th>TMC Iter.</th>
<th>TMC Stop ctr.</th>
<th>TMC Neigh.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>20</td>
<td>10</td>
<td>30</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>20</td>
<td>10</td>
<td>30</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>15</td>
<td>50</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>D</td>
<td>30</td>
<td>15</td>
<td>50</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Average executed score after 20 iterations:

<table>
<thead>
<tr>
<th>Settings</th>
<th>PIX</th>
<th>TMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>135.78</td>
<td>712</td>
</tr>
<tr>
<td>A</td>
<td>136.13</td>
<td>1,113</td>
</tr>
<tr>
<td>B</td>
<td>136.29</td>
<td>1,013</td>
</tr>
<tr>
<td>C</td>
<td>138.17</td>
<td>1,799</td>
</tr>
<tr>
<td>D</td>
<td>142.06</td>
<td>3,116</td>
</tr>
</tbody>
</table>

Relative change (in %) against base case:

<table>
<thead>
<tr>
<th>Settings</th>
<th>PIX</th>
<th>TMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>0.27</td>
<td>0.27</td>
</tr>
<tr>
<td>A</td>
<td>2.71</td>
<td>2.71</td>
</tr>
<tr>
<td>B</td>
<td>62.36</td>
<td>62.36</td>
</tr>
<tr>
<td>C</td>
<td>152.71</td>
<td>152.71</td>
</tr>
<tr>
<td>D</td>
<td>337.66</td>
<td>337.66</td>
</tr>
</tbody>
</table>

PlanomatX is more sensitive to parameter settings than the stand-alone TimeModeChoicer. Number of iterations and neighbourhood size seem to influence PlanomatX’s optimization performance. Joint parameter variation with the TimeModeChoicer yields some clear insights: reduction of number of TimeModeChoicer iterations is completely positive (see setting E in figure 4.22). Further runtime reductions could be gained by reducing the number of PlanomatX iterations.

4.6.2.5 Summary
4.6.3 Greater Zurich scenario

The greater Zurich scenario\footnote{Full name is greater Zurich scenario 10\% diluted. Descriptions draw from Balmer \textit{et al.} (2008b), chapter 7.} comprises a set of 172,598 agents, a 10\% random draw from those agents whose initial demand routes cross a 30 km circle around Zurich’s city centre. The road network is represented by a model network of 60,000 directed links and 24,000 nodes. There are 1.3 million home locations and more than 380,000 out-of-home locations. Available transport modes are car, public transport, bicycle, and walk. Eleven activity types are modelled (outer and inner home, two work types, five education/school types, leisure, and shopping). Agents’ initial schedules have been assembled according to the procedure described in section 2.3.1.1.

4.6.4 Test VI: Test of PlanomatX on greater Zurich scenario

Test VI repeats test IV, but is a “real-world” simulation run on the greater Zurich scenario. Instead of the initially set utility function, the empirically estimated and calibrated utility function from chapter 6 is applied. Further parameter settings are identical to those from test IV.
4.6.4.1 Development of utility scores, trip travel distances and times

Figure 4.24 shows the development of utility scores, trip travel distances and times. Like on the chessboard scenario, PlanomatX achieves a significantly better utility score (7.75 points) than the base test (6.23 points), a plus of 38%\(^{10}\). While average trip travel distance remains almost fixed in the base test, it falls to 5.0 km for PlanomatX. Number of activities per schedule increases (see below) and PlanomatX is able to locate additional activities “on the way” between the existing ones. Average trip travel time falls significantly for both tests. Agents adapt to the traffic situation and move their trips into off-peak hours. Quite interesting is the bend at iteration 11 in the base test’s curves. Average trip travel time suddenly falls, utility rises. One can assume that some heavy traffic congestion is cleared but we have not investigated exact reasons for the bend.

4.6.4.2 Comparison with reported and observed data

Beyond the development of utility, as well as of travel distances and times over the course of the simulation, final activity and trip characteristics should be analyzed and compared against reported and observed data. Note that the utility function has been calibrated for the use of schedule recycling (see chapter 5). Due to their heuristic nature, differences in PlanomatX’s and schedule recycling’s optimization results exist and, thus, PlanomatX’s results do not match reported and observed data as well as schedule recycling’s results will in test VIII (see page 117 et sqq.):

- Activities: Average activity chain length is 5.52 activities for PlanomatX (see fig-

---

\(^{10}\)Subject to base score of 2.28 points at iteration 0.
Figure 4.24: Test VI: Development of average executed schedule utility, trip distance and trip travel time.

It is 4.92 for the base test which is, by definition, equal to the initial demand’s average activity chain length. Both PlanomatX and base test feature longer average activity chains than reported by Microcensus (4.65 activities). Some positive difference is desired since people tend to report fewer activities than really undertaken. 5.52 activities a day seems at the upper limit of the acceptable range. Additional activities are mainly reflected by the increased daily frequency of the leisure activity type. Overestimation of leisure is substantially higher than it will be for schedule recycling. For PlanomatX, average leave and return times from/to home are about 40 min late. Except for leisure, average activity type durations are lower than reported (see figure 4.25a, upper chart). Simulated time spent on performing activities is 24 min higher than reported.
PlanomatX’s twenty most frequent activity chains approximate the twenty most frequent activity chains reported by Microcensus (see table 4.2). However, specific numbers of occurrences often differ. For instance, “home-work-leisure-home” is PlanomatX’s preferred activity chain. It is “home-work-home” according to Microcensus. The base test,
Table 4.2: Test VI: Analysis of 20 most frequent activity chains

<table>
<thead>
<tr>
<th>Activity chain</th>
<th>Base test</th>
<th>PlanomatX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of occurrences</td>
<td>Relative</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>22995</td>
<td>13.32%</td>
</tr>
<tr>
<td>home leisure home leisure home</td>
<td>12521</td>
<td>7.25%</td>
</tr>
<tr>
<td>home leisure work leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
<tr>
<td>home leisure leisure leisure home</td>
<td>2764</td>
<td>1.60%</td>
</tr>
</tbody>
</table>

by definition, matches the reported activity chains quite well. PlanomatX performs very well with regard to the diversity of activity chains. While Microcensus reports 560 different activity chains, PlanomatX increases the number to 4,397 different chains. A higher number is beneficial, as it increases the model’s entropy.

- **Travel**: In line with the increased number of activities, average number of trips per schedule is 4.52 for PlanomatX and 3.90 for the base test (see figure 4.25a, lower chart). Both are again higher than the reported number of 3.65 trips. PlanomatX slightly overestimates car and public transport, and slightly under-estimates bike. Agents walk nearly twice as often as reported by Microcensus. This is because PlanomatX reduces the average trip travel distance more rigorously than schedule recycling. Walk is attractive at short distances and gains in mode share against the other modes. We will see in chapter 5 that this effect fades with schedule recycling. The base test performs just the other way around than PlanomatX; only walk (0.82 trips) roughly matches the reported figure (0.88 trips). All other mode figures seem to differ more significantly.

Comparison of average travel distances and times is quite interesting. For PlanomatX and base test, simulated total distances per schedule (22.7 km and 24.8 km) are lower than the reported (41.5 km). This is, at least partly, due to a weighting bias of Microcensus respondents compared to commuter data input to the initial demand definition. As a consequence, if agents’ primary locations are “too close” to each other, it is very hard to add distance to the schedules - unless traffic situation requires detours (see base test), or one rates distance very positively in the utility function so that travel as a whole becomes pos-
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

Figure 4.26: Test VI: Reported and simulated number of trips and mode shares (calibrated utility function)

(a) Microcensus: reported number of trips and mode shares by distance

(b) Microcensus: reported cumulated mode shares by distance

(c) Base test: simulated number of trips and mode shares by distance

(d) Base test: simulated cumulated mode shares by distance

(e) PlanomatX: simulated number of trips and mode shares by distance

(f) PlanomatX: simulated cumulated mode shares by distance


However, this is problematic. The router module requires a function that scores a fast (≈ short) route better than a slow (≈ long) route. A solution to the problem may be the revision of agents’ primary locations or an attraction-driven location choice module. The latter may make far-off secondary locations more attractive than near-by options,

This is different from the calibrated utility function where car travel is associated with a positive distance parameter but car travel as a whole is still negative.
Figure 4.27: Test VI: Observed and simulated traffic counts (calibrated utility function)

(a) Total of 115 counting stations within 10km circle around Zurich city centre

(b) Counting station Zurich Hardtstrasse, direction NE towards Bucheggplatz (code 106311)

when they are, for instance, uncrowded (see outlook, chapter [7]). Beyond the general loss of distance, only public transport gains in distance compared to the initial demand. However, it is still below the public transport figure reported by Microcensus for both PlanomatX and base test.

Figure 4.26 emphasises trip numbers and mode shares by distance. The base test develops a very sharp peak in trip numbers at 2-5 km. It has virtually lost the bump at 0.2-0.5 km. PlanomatX overestimates the bump at 0.2-0.5 km. The peak at 2-5 km flattens out. The analysis of mode shares by distance confirms that the base test underestimates car mode. Especially in the medium distance classes, public transport dominates car.

Note that the distance classes 0km and 0-0.1km should be ignored in the analysis. Their Microcensus figures are implausible. The high simulation share of car mode in the class of 0km is due to the sub-tour based mode choice (see section 4.3).
Bike has too strong a share in the classes from 1-5 km. The increasing share of walk mode toward the long distance is implausible, but it is irrelevant, since trip numbers are very small. It becomes evident for PlanomatX that walk mode holds massive shares in the two classes up to 0.1 km. Given that we may ignore them, PlanomatX has quite nicely reproduced the walk mode. Nonetheless, the bike share is too small. Public transport and car look very similar to the base test for the longer-distance classes.

Figure 4.27 compares simulated and observed car traffic counts. Both charts show that the base test replicates the observed traffic counts slightly better than PlanomatX. PlanomatX produces some individual peaks, most of all at 4, 5 and 8 pm.

### 4.6.4.3 Runtime

Both PlanomatX and base test were run on a high-end computer with 16 Itanium-2 Dual Core processors. The overall run time of the PlanomatX run was 7 days and 3 hours. Each MATSim iteration took about 204:35 min (see figure 4.28). It splits into 6:30 min of traffic flow simulation and 195:30 min of PlanomatX replanning (0.676 sec per agent). The overall run time of the base test was only 11:32 hours. Here, each MATSim iteration took about 13:40 min splitting into 6:30 min of traffic flow simulation and 4:24 min of replanning run time (0.015 sec per agent). One must conclude that PlanomatX’s better scoring results come at a high run time cost; while scoring results have improved by factor 1.38,

- overall runtime has nearly increased by factor 15, and
- pure replanning runtime has increased by factor 45.
4.6.4.4 Summary

PlanomatX achieves a higher utility score than the base test (factor +1.8). However, the utility increase still comes with a massive runtime cost (factor +45). Moreover, neither PlanomatX nor the base test reproduce the reported/observed data satisfactorily. Activity and trip characteristics differ from those reported by Microcensus or collected through traffic counts. We will see in test VIII that schedule recycling will largely relieve these drawbacks since the new utility function has been calibrated for schedule recycling.

4.7 Discussion

TimeModeChoicer is an alternative to Planomat, featuring a similar optimization functionality. However, it runs about six times faster. It reduces the runtime of activity timings optimization and mode choices to about 6 msec per agent. TimeModeChoicer and Planomat achieve the same utility level and strongly resemble each other in output results (see figure 4.8). Nevertheless, one must bear in mind that TimeModeChoicer requires more constraints than Planomat, the most important being minimum activity durations and 24 hours schedule duration.

TimeModeChoicer can be run as a MATSim stand-alone replanning strategy. However, it is particularly designed for integration with PlanomatX and helps reduce PlanomatX’s overall runtime. Still, PlanomatX consumes substantially more runtime than the base test as tests V and VI have shown. A simulation runtime of more than a week for the greater Zurich scenario is impractical and prevents broad application. Moreover, PlanomatX does not reproduce reality satisfactorily. A number of activity and trip characteristics have differed from those reported by Microcensus and observed in traffic counts. In chapter 5, we will present a concept of schedule recycling that addresses both problems:

- Schedule recycling exploits further potentials of runtime reduction. It avoids running PlanomatX for each agent individually but rather re-uses schedules of optimized agents for other non-optimized agents.

- MATSim’s new utility function has been calibrated for the application of schedule recycling (see chapter 6). We will see in test VIII that considerably more realistic activity and travel characteristics can be achieved.
Chapter 5

Schedule recycling

Schedule recycling is a concept that re-uses optimized agents’ schedules for non-optimized agents. Aiming for reduced runtimes, it avoids running PlanomatX for each agent individually.

5.1 Problem formulation

PlanomatX successfully optimizes activity-travel schedules, but leads to disproportional runtimes. The activity chain, as a new dimension of MATSim’s co-evolutionary learning process, is one main reason for heavily increased runtimes. When analyzing table 4.2, it is evident that many agents share the same, or similar, activity chains even if the activity chain is flexible. A set of 4,397 different activity chains is sufficient to cover the scenario’s 170,000 plus agents’ activity chains. The twenty most frequent activity chains already cover more than 50% of all agents. Microcensus reports 560 different activity chains, with the twenty most frequent activity chains covering more than 45% of all travellers. To reduce runtimes, a way is sought to re-use, or “recycle”, schedules of optimized agents for other non-optimized agents, without running PlanomatX for each agent individually. The following problems emerge: which agents should be optimized individually, and which agents should be provided with a recycled schedule? Which elements of a schedule may be recycled? Which rule, or metric, governs how agents should be assigned with those elements? Finally, what could the process flow of schedule recycling look like?

5.2 Solution algorithm

The schedule recycling algorithm (see figure 5.1) is an alternating sequence of optimizing agents individually (“PlanomatX”) and assigning those optimized agents’ schedules to non-
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

Figure 5.1: Schedule recycling process flow

- Receive all agents to be replanned from controller, initialize algorithm
- "PlanomatX": Optimize a random set of agents individually (e.g., 100 agents)
- Find optimal distance metric for continuous agents attributes
- "Assignment": Assign a random set of non-optimized agents (e.g., 500 agents) with schedules from the optimized agents
- "PlanomatX": Individually optimize those agents for which no schedule was assignable in step D
- "Assignment": Assign all remaining non-optimized agents with schedules from the optimized agents
- "PlanomatX": Individually optimize those agents for which no schedule was assignable in step F
- Return all agents to controller

Every 10 MATSim iterations only
See figure 3.2
See figure 5.2
See figure 3.2
See figure 5.2

optimized agents ("Assignment"). The algorithm features three core innovations: assignment module (steps D and F), definition of the distance metric (step C), and the revolving sequence of PlanomatX (steps B, E and G) and assignment module (steps D and F). To clarify, first the assignment module will be explained, followed by the arguments of the process flow rationale and definition of the distance metric.

5.2.1 Assignment module (steps D and F)

The assignment module provides non-optimized agents with recycled schedules. For any non-optimized agent, the module defines a best matching schedule from a given set of individually optimized agents, assigns the non-optimized agent with that schedule’s activity chain, and finalizes the schedule’s location, route, and mode choices, as well as activity timings. Figure 5.2 illustrates the assignment module’s process flow.

Initialization of the assignment module (step D/F1)
The assignment module receives the non-optimized agent’s schedule, to be assigned with a recycled schedule.
Best-match operator (steps D/F2 to 5)

After initialization, the assignment module defines which agent’s schedule from a given set of individually optimized agents matches the non-optimized agent best. Definition of best matching schedule relies on three stages that loop through the set of individually optimized agents:

- **Check of activity type constraints (step D/F2):** This first check verifies whether the schedule of an individually optimized agent contains all primary activity types of the non-optimized agent. The check also verifies that the individually optimized agent’s schedule does not contain activity types ineligible to the non-optimized agent. An example would be “work” for a 10-year old child or “primary education” for a 50-year old adult (see section 3.2.2.1 paragraph “Insertion operator (step B1.1)”, for full list of rules).

- **Check of discrete agents attributes (step D/F3):** Some agent attributes may prevent an agent from receiving a recycled schedule from an individually optimized agent. Four rules are possible:
– An agent without a driver’s license may not adopt a schedule containing car trips.
– An agent without a car may not adopt a schedule containing car trips.
– A male agent’s schedule may only be recycled for another male agent, or vice versa.
– An employed agent’s schedule may only be recycled for another employed agent, or vice versa.

Only the last rule is currently enforced. The first and second rules cannot be backed up by the current level of input information, since car mode is not differentiated by driving and ride-sharing. The third rule seems too restrictive at this time.

• **Check of continuous agents attributes’ distance (step D/F4)**: From the set of individually optimized agents, it is likely that several schedules will pass the first two checks. This third check determines which of these schedules matches the non-optimized agent best. It refers to the distance, or similarity, between continuous attributes of optimized agent \(i\) and non-optimized agent \(j\). The distance \(d\) is calculated as follows:

\[
d_{i,j} = \sum_{k=1}^{n} |x_{i,k} - x_{j,k}| \cdot \delta_k
\]

where \(\delta_k\) is the weight coefficient of the continuous attribute \(k\). Current data availability allows to use three continuous attributes:

– agents’ geographic distances between their primary activities (e.g., home-work-home),
– agents’ home coordinates\(^1\), and
– agents’ ages.

The definition of multidimensional distance metric \(\Delta = \{\delta_1; \ldots; \delta_k; \ldots; \delta_n\}\) will be offered below.

• **Selection of best-matching schedule (step D/F5)**: Having iterated through all optimized agents, the assignment module selects, in step 5, the schedule that passed the two first checks and whose agent features the highest similarity (= lowest distance) with the non-optimized agent.

**Assignment operator (step D/F6)**

This operator assigns the non-optimized agent the schedule selected in the previous step: it copies the activity chain of the selected schedule and pastes it into the non-optimized agent’s new schedule. At this time, the new schedule holds only the activity chain.

\(^1\)This works surprisingly well as a rough approximation of whether agents live in urban areas or in the country.
Customization operator (step D/F7)
The customization operator adds all further agent-specific information to the schedule:

- **Location choice**: The operator updates home location as well as locations of the new activity chain’s other primary activities. The operator conducts a full location choice for all remaining secondary activities (see section 3.2.4.1).

- **Route choice**: Shortest routes between locations are found using MATSim’s standard router module (see section 3.2.4.2).

- **Optimization of activity timings and mode choices**: Finally, the TimeModeChoicer is run to define optimal activity timings and mode choices (see section 3.2.4.3).

It becomes evident that location, route, and mode choices, as well as optimization of activity timings, need to be run only once for an agent assigned a recycled schedule, unlike PlanomatX, where the steps need to be conducted for each of the agent’s neighborhood solutions of the Tabu Search (“inner loop”, see section 3.2.4). The lower need for optimization steps allows schedule recycling’s lower runtime.

5.2.2 Rationale of revolving process flow

Let us highlight the role of the three stages in selecting the best matching schedule; the first two checks are responsible for the alternating sequence of optimizing agents individually (“PlanomatX”) and assigning schedules of optimized agents to non-optimized agents (“Assignment”). An a-priori search for optimized schedules passing the checks for the non-optimized agents’ group would be very cumbersome. It is much easier just to select some agents randomly, optimize their schedules individually (step B) and see whether these schedules pass the checks for the non-optimized agents (step F). Some non-optimized agents will remain for which no schedule is assignable, and they must be optimized individually (step G).

This sequence would work perfectly well, but our algorithm refines it, introducing steps D and E. Step D checks - for a limited number of non-optimized agents - whether the optimized step B schedules pass the checks for these non-optimized agents. Those non-optimized agents (for which no schedule is assignable) are handed over to step E. Step E optimizes them individually and adds them to the list of optimized step B agents. In this form, they are available for step F. Step’s F assignment now produces many fewer non-optimized agents for which no schedule is assignable. As a result, runtime of step G is shortened, and runtime reduction will probably be much higher than the runtime increase from step E. In theory, one may introduce more instances of steps D and E. In the most advanced case, every non-optimized agent for which no schedule is assignable is immediately optimized individually and added to the list.
of optimized agents. However, this makes parallelization of the algorithm extremely hard. The current algorithm setup is a reasonable trade-off.

Only the number of agents to be optimized individually in step B must be configured externally. The optimal setting is a trade-off between runtime consumption and schedule entropy. Simulation runtime increases with the number of individually optimized schedules. A high number of individually optimized schedules, on the other hand, expands the set of schedules to select from, or the set of activity chains, respectively. Thus, an optimal setting is highly dependent on the specific scenario.

- In the small chessboard scenario, 5 agents out of an iteration’s 32 replanning agents have emerged as a reasonable number of individually optimized agents. Any first agent’s schedule ensures that all recycled schedules fulfil the primary activity type constraint of check 1 (step D/F2), namely that every agent must perform “home” and “work”. The other four schedules add entropy to the schedule set.

- In the large-scale greater Zurich scenario, a setting of 100 agents out of an iteration’s approximately 17200 replanning agents yields very satisfying results. 100 agents offer enough upfront schedule variability to cover most recycled agents’ primary activity type and personal attribute constraints (step D/F2 and 3). On the other side, the number of agents is still limited, so that simulation runtime does not increase too much.

5.2.3 Definition of distance metric (step C)

Distance metric $\Delta$ of check 3 is central to the quality of the assignment module. In fact, check 3 conducts a kind of cluster analysis. It clusters non-optimized agents around (assignable) optimized agents and, in each cluster, non-optimized agents are assigned the schedule of the optimized agent. In conventional cluster analysis, the distance metric $\Delta$ is given (Euclidean distance, Hamming distance, etc.) and attention is paid to the most efficient way of clustering.

In our case, the clustering method is quite straightforward (see above) but the distance metric $\Delta$ is unknown. For example: given the three earlier attributes (geographic distance between an agent’s primary activities, an agent’s home coordinates and an agent’s age), we want to know how much each distance, or similarity, between the attribute values of any two agents contributes to the similarity $d_{i,j}$ of those two agents’ $i$ and $j$ optimized activity chains. If agents’ distances between their primary activities were twice as important than the other two

---

2We have not conducted performance tests for this parameter as we did for PlanomatX and TimeModeChoicer settings. Agents are quite homogeneous in the chessboard scenario (similar knowledge with similar primary activity types) and performance tests would not be meaningful.

3Let us remember that MATSim generally allows 10% of all agents to replan per iteration, here 32 out of the scenario’s overall 324 agents.
attributes, the distance metric would look like $\Delta = \{\delta_{\text{distance\_primacts}}; \delta_{\text{home\_coordinates}}; \delta_{\text{age}}\} = \{2; 1; 1\}$. If all attributes were equally important, the distance metric would look like $\Delta = \{\delta_{\text{distance\_primacts}}; \delta_{\text{home\_coordinates}}; \delta_{\text{age}}\} = \{1; 1; 1\}$, and so forth.

Solving for the distance metric $\Delta$, we have developed a “reverse clustering” approach. The solution algorithm forms two sub-groups of agents. Group 1 is a group of optimized agents, ideally the group optimized agents of step B (see above, 100 agents for the Greater Zurich scenario). Group 2 is a group of randomly chosen non-optimized agents. The number of agents
must be given externally, and should be four to five times higher than the number of agents in group 1 (i.e., 20 agents in the chessboard scenario, 500 agents in the Greater Zurich scenario).

After having individually optimized the agents of group 1 through PlanomatX, the algorithm provides the non-optimized agents with optimized schedules according to an arbitrary distance metric, for instance $\Delta_t = \{\delta_1; \ldots; \delta_k; \ldots; \delta_K\} = \{1;\ldots;1;\ldots;1\}$. The assigned schedules will sum up to a certain utility $U_t = U_{\{1;\ldots;1;\ldots;1\}}$. The algorithm repeats the test assignment for $n_I$ times. Each time, a varying set of distance metrics $\Delta_t$ is checked for the utility $U_t$ to which they would lead. The variation is done through increasing/decreasing all, except one, coefficient of the current metric by an offset value (e.g., 0.5; see figure 5.3). One coefficient must be a reference coefficient with a fixed anchor value for instance $\delta_{age} = 1$. The alternative, for which the utility becomes highest, is chosen as the base metric for the following variation of coefficients. A variation undoing previous metrics is forbidden to ensure an efficient search of the solution space. Finally, the algorithm selects the distance metric $\Delta_s$ for which the utility of the test agents’ schedules is highest across all $x$ times of the test assignment, i.e. $U_s > U_t$, $\forall t \neq s$.

The distance metric must be determined when the schedule recycling is first applied, generally at the beginning of MATSim’s first iteration’s replanning step. Over the course of simulation, traffic loads may change and the metric definition should be re-verified. This happens every ten MATSim iterations with $n_{II}$ test assignments ($n_{II} < n_I$).

5.3 Test results of schedule recycling

We have run two tests to analyze the performance of schedule recycling:

- Test VII compares schedule recycling with PlanomatX and MATSim’s existing replanning strategies on the chessboard scenario.

- Test VIII does the same for the greater Zurich scenario and serves as calibration test for the new utility function.

---

4 This is comparable to the concept of $n - 1$ mode choice constants ($n$ = number of available modes). We ignore the fact that the most correct value of the reference coefficient may be 0.

5 Note that we ultimately search for a metric that ensures the recycled schedules are a best possible replication of the individually optimized activity chains. Selecting the distance metric $\Delta_s$ with the highest utility $U_s$ is not exactly the same. On the other side, individually optimized activity chains should result in a maximum utility and, thus, the procedure seems justified.
5.3.1 Test VII: Test of schedule recycling on chessboard scenario

Test VII analyzes the performance of schedule recycling on the chessboard scenario. PlanomatX and MATSim’s existing replanning strategies serve as benchmarks (see test IV). The following settings are applied:

- All MATSim settings are those from test IV, but schedule recycling is the 10% replanning strategy rather than PlanomatX.

- Schedule recycling settings:
  - *Individual optimization of agents*: 5 agents are optimized individually in step B.
  - *Metric definition*: Step C runs 20 variation steps in MATSim’s iteration 1 \((n_I = 20)\), and 5 iterations afterward \((n_{II} = 5)\). The metric is defined based on attributes distances between agents’ primary activities, agents’ home coordinates, and agents’ ages.
  - *Test assignment*: 10 agents are assigned an optimized schedule in step D.

- TimeModeChoicer settings:
  - *TimeModeChoicer called by PlanomatX*: When called by PlanomatX, TimeModeChoicer runs with a neighbourhood size of 10 solutions, with 5 outer loop iterations, and a stopping criterion of 5 iterations\(^6\). These settings are in line with the findings of test V, section 4.6.2.4, scenario (I).
  - *TimeModeChoicer called by assignment module of schedule recycling*: When called by the assignment module of the schedule recycling, the one-time call requires a more extensive optimization. Therefore, TimeModeChoicer runs with 30 outer loop iterations, c.p. This setting is based on the findings of test III, section 4.5.2.2, base case.

5.3.1.1 Development of utility scores, trip travel distances and times

The utility curves of schedule recycling and PlanomatX almost overlap (see figure 5.4). Schedule recycling achieves a final utility level of 135.83 points. This is marginally better than PlanomatX’s final utility score of 135.78 points. Both strategies perform better than the base test (117.53 points). Schedule recycling results in both a slightly lower average trip distance and time. This is due to the increased average activity chain length (5.41 vs. 5.30; see previous section); agents perform more activities, hence the distance between the activity locations falls.

\(^6\)This means that 5 iterations are run. The stopping criterion has no effect.
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

Figure 5.4: Test VII: Development of average executed schedule utility, trip distance and trip travel time

![Graph showing utility, travel distance, and travel time over MATSim iterations]

Most important is that both schedule recycling’s and PlanomatX’s trip travel time curves fall substantially below that of the base test.

5.3.1.2 Metric coefficients

Check 3 (step D/F4) determined the distance between agents, based on three attributes: agents’ primary activities, agents’ home coordinates, and agents’ ages. Figure 5.1 shows development
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

Table 5.1: Test VII: Development of distance metric coefficients

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>MATSim iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance between agent’s primary activities</td>
<td>1.0 0.5 2.5 3.0 3.0 3.5 4.0 4.0 4.5</td>
</tr>
<tr>
<td>Agent’s home location coordinates</td>
<td>2.5 1.5 2.0 3.0 3.0 4.0 4.5 4.0 4.5</td>
</tr>
<tr>
<td>Agent’s age</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
</tr>
</tbody>
</table>

of attribute coefficients over the MATSim iterations. Difference in agents’ ages is the reference attribute with the anchor coefficient $\delta_{\text{age}} = 1$. The other two attributes gain importance over the iterations and both end up being 4.5 times more important than the difference in agents’ ages. This is logical since

- geographical attributes may well be more relevant to the choice of activity chain than age,
- the chessboard utility function does not take age into account.

The second point implies that if the age coefficient was not fixed to $\delta_{\text{age}} = 1$, it should be $\delta_{\text{age}} = 0$, or dropped from the set of metric coefficients completely. However, inclusion of the age coefficient has confirmed that the distance metric algorithm is correct, since other coefficients developed as expected.

5.3.1.3 Initial demand and optimized schedules

Schedule recycling and PlanomatX perform similarly for optimized activity chains (see figure 5.2). Average activity chain lengths are 5.41 and 5.30. Most of schedule recycling’s optimized activity chains have also been optimal with PlanomatX. The five most frequently occurring chains are the same for both schedule recycling and PlanomatX. Some differences exist only in the specific numbers of chains’ occurrences: PlanomatX tends to optimize strongly toward the chain “home-work-shopping-leisure-home” (102 occurrences). Schedule recycling balances out the number of occurrences among chains.

Examining example agent 1, figure 5.5 shows that schedule recycling results in one additional shopping activity for the agent. While PlanomatX lets the agent spend 2:08 hours at a single shopping activity, schedule recycling splits the activity into two shopping activities of 30:00 min and 1:29 hours. All further activity durations remain quite stable. Shopping and leisure activities have changed locations (not shown in figure), but the agent still uses public transport throughout the day.
Table 5.2: Test VII: Analysis of activity chains

<table>
<thead>
<tr>
<th>Activity chain</th>
<th>Base test</th>
<th>PIX/TMC</th>
<th>Schedule recycling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of occurrences</td>
<td>Relative</td>
<td>Number of occurrences</td>
</tr>
<tr>
<td>home work shopping home</td>
<td>163</td>
<td>50.3%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>161</td>
<td>49.7%</td>
<td>11</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>102</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>69</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>41</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>31</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>24</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>12</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>7</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>5</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>4</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>3</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>2</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>2</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>1</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>1</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>home work leisure home</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
</tbody>
</table>

Average number of activities: 4.00, 5.30, 5.41

Figure 5.5: Test VII: Initial demand schedule and optimized schedules of agent 1 (for sake of simplification, not showing activity locations and travel routes)

5.3.1.4 Runtime

The overall runtime of the schedule recycling test was 9:50 min. This is more than four times faster than the PlanomatX test runtime (42:20 min), and less than twice as high as base test runtime (5:17 min). Figure 5.6 shows runtime shares per MATSim iteration. Schedule recycling’s
first MATSim iteration takes 17.4 sec. In this iteration, the distance metric algorithm is run with \( n_I = 20 \) metric variations. The algorithm takes 12.0 sec. A standard schedule recycling iteration without the distance metric algorithm takes 5.4 sec. The individual optimization of schedules (PlanomatX) runs for about 2.2 sec. The assignment module runs for about 0.8 sec. On average, schedule recycling’s replanning runtime (6.0 sec) is almost six times faster than PlanomatX’s replanning runtime (23.0 sec), and about nine times higher than the base test’s replanning time (1.1 sec).

5.3.1.5 Summary

Schedule recycling achieves the same utility level as PlanomatX, but runs about four times faster, and pure re-planning runtime is about six times faster. The optimized activity chains largely overlap, but some differences exist in their number of occurrences: PlanomatX leads to a few, very frequent activity chains, while schedule recycling results in a set of chains with a more balanced number of occurrences.

5.3.2 Test VIII: Test of schedule recycling on greater Zurich scenario

Test VIII repeats test VII, but is a “real-world” simulation run on the greater Zurich scenario. As with the base test and PlanomatX, MATSim’s new utility function, described in chapter 6

\[ \text{Based on the greater Zurich scenario, schedule recycling runtimes will be discussed in more detail in section 5.3.2.4.} \]
Table 5.3: Test VIII: Development of distance metric coefficients

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>MATSim iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Distance between agent’s primary activities</td>
<td>3.0</td>
</tr>
<tr>
<td>Location of agent’s home coordinates</td>
<td>2.0</td>
</tr>
<tr>
<td>Agent’s age</td>
<td>1.0</td>
</tr>
</tbody>
</table>

is used. Utility function has been calibrated based on this test; this will manifest itself in a substantially better fit of simulated and reported/observed data. All test settings are those from test VI, except that schedule recycling is applied as replanning strategy.

5.3.2.1 Development of utility scores, trip travel distances and times

As figure 5.7 shows, schedule recycling achieves a higher utility score (7.34 utility points) than the base test (6.23 utility points). Measured against the initial utility score of 2.28 points at iteration 0, this is a plus of 28%. However, schedule recycling falls 7% below PlanomatX’s final utility score (7.75 utility points). This is different from the chessboard scenario where schedule recycling finished at the same utility level as PlanomatX. The most obvious reason is the difference in average trip travel distance between schedule recycling and PlanomatX. As opposed to PlanomatX, schedule recycling has difficulties lowering the distance, because location choice for re-used schedules is not as efficient as that of individually optimized schedules. The effect is not evident on the chessboard scenario, which is more limited in space and offers less opportunity to vary agents’ secondary location choices. In line with the longer average trip travel distance, schedule recycling’s average trip travel time is also longer than PlanomatX’s average trip travel time.

5.3.2.2 Metric coefficients

Figure 5.3 shows development of the attribute coefficients over the MATSim iterations. Agents’ age is the reference attribute with the anchor coefficient $\delta_{\text{age}} = 1$. The other two attributes gain importance over the iterations. Agents’ distance between their primary activities is ultimately rated 1.5 times more importantly than age, and distance between agents’ home coordinates 4.0 times more importantly. To summarise, development of coefficients is less smooth than in the chessboard scenario, possibly due to the more complex nature of the greater Zurich scenario.
5.3.2.3 Comparison with reported and observed data

While differences in the optimization results between PlanomatX and schedule recycling are unfortunate, parameters of the new utility function have been calibrated for the use of schedule recycling (see chapter 6), because it will probably be used more often than PlanomatX, due to its better runtime. This test represents the final calibration run and, hence, reproduces reality substantially better than the previous PlanomatX test. Simulated and reported/observed data generally match well:

- Activities: The calibrated utility function leads to an average activity chain length of 4.93 activities (see figure 5.8a, lower chart). This is slightly higher than the average length
Table 5.4: Test VIII: Analysis of 20 most frequent activity chains (for ease of illustration, only Microcensus, PlanomatX and schedule recycling shown)

<table>
<thead>
<tr>
<th>Activity chain</th>
<th>Microcensus weighted</th>
<th>PlanomatX</th>
<th>Schedule recycling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of occurrences</td>
<td>Number of occurrences</td>
<td>Number of occurrences</td>
</tr>
<tr>
<td></td>
<td>Relative</td>
<td>Relative</td>
<td>Relative</td>
</tr>
<tr>
<td>home-work-leisure-home</td>
<td>98.03 1.96%</td>
<td>22995 13.32%</td>
<td>44566 25.82%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>81.84 1.63%</td>
<td>12521 7.25%</td>
<td>14332 8.30%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>779.96 15.56%</td>
<td>6943 4.02%</td>
<td>8426 4.88%</td>
</tr>
<tr>
<td>home-work-home</td>
<td>164.29 3.28%</td>
<td>4747 2.75%</td>
<td>6462 3.74%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>127.13 2.54%</td>
<td>2180 1.26%</td>
<td>5014 2.91%</td>
</tr>
<tr>
<td>home-work-home</td>
<td>32.49 0.65%</td>
<td>8438 4.89%</td>
<td>4492 2.60%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>203.15 4.05%</td>
<td>4406 2.55%</td>
<td>4079 2.36%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>8.86 0.18%</td>
<td>4664 2.70%</td>
<td>3518 2.04%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>13.83 0.28%</td>
<td>1216 0.70%</td>
<td>3029 1.75%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>10.16 0.20%</td>
<td>6785 3.93%</td>
<td>2587 1.50%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>104.98 2.09%</td>
<td>1491 0.86%</td>
<td>2552 1.48%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>155.24 3.10%</td>
<td>4731 2.74%</td>
<td>2357 1.37%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>68.49 1.37%</td>
<td>2033 1.18%</td>
<td>2199 1.27%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>1.12 0.02%</td>
<td>1402 0.81%</td>
<td>1742 1.01%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>5.27 0.11%</td>
<td>1030 0.60%</td>
<td>1541 0.89%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>10.13 0.20%</td>
<td>2305 1.34%</td>
<td>1533 0.89%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>0.00 0.00%</td>
<td>4637 2.69%</td>
<td>1389 0.80%</td>
</tr>
<tr>
<td>home-leisure-home-home</td>
<td>0.00 0.00%</td>
<td>1583 0.92%</td>
<td>1361 0.75%</td>
</tr>
<tr>
<td>home-education-home-home</td>
<td>8.83 0.18%</td>
<td>1771 1.03%</td>
<td>1312 0.76%</td>
</tr>
</tbody>
</table>

Average number of activities: 4.65
Number of different activity chains: 560

- Travel: In line with the slightly increased number of activities, the average number of trips per schedule (3.93 trips) is somewhat higher than the reported number (3.65 trips).
Figure 5.8: Test VIII: Reported and simulated activity and travel statistics (for ease of illustration, only observed traffic counts and schedule recycling counts shown)

(a) Activity statistics

<table>
<thead>
<tr>
<th></th>
<th>Microcensus simulated</th>
<th>Schedule recycling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leave</td>
<td>9:04</td>
<td>9:07</td>
</tr>
<tr>
<td>Return</td>
<td>17:56</td>
<td>18:04</td>
</tr>
<tr>
<td></td>
<td>2.08</td>
<td>2.03</td>
</tr>
<tr>
<td></td>
<td>4.42</td>
<td>4.12</td>
</tr>
<tr>
<td></td>
<td>3.57</td>
<td>3.31</td>
</tr>
<tr>
<td></td>
<td>1.55</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>0.44</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>22:48</td>
<td>22:53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Inner home</td>
<td>0.50</td>
<td>0.44</td>
</tr>
<tr>
<td>Work</td>
<td>0.88</td>
<td>0.78</td>
</tr>
<tr>
<td>Education</td>
<td>0.20</td>
<td>0.24</td>
</tr>
<tr>
<td>Leisure</td>
<td>0.74</td>
<td>1.19</td>
</tr>
<tr>
<td>Shopping</td>
<td>0.37</td>
<td>0.28</td>
</tr>
<tr>
<td>Total per schedule</td>
<td>4.65</td>
<td>4.93</td>
</tr>
</tbody>
</table>

(b) Travel statistics

<table>
<thead>
<tr>
<th></th>
<th>Microcensus simulated</th>
<th>Schedule recycling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14.1</td>
<td>6.2</td>
</tr>
<tr>
<td></td>
<td>20.7</td>
<td>16.9</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>41.5</td>
<td>24.9</td>
</tr>
</tbody>
</table>


All mode-specific figures match quite well, although a slight preference is seen in car mode. Beyond the general loss of distance, only public transport gains in distance compared to the initial demand. However, it is still slightly below the public transport figure reported by Microcensus.
Figure 5.9: Test VIII: Reported and simulated number of trips and mode shares

(a) Microcensus: reported number of trips and mode shares by distance

(b) Microcensus: reported cumulated mode shares by distance

(c) Base test: simulated number of trips and mode shares by distance

(d) Base test: simulated cumulated mode shares by distance

(e) PlanomatX: simulated number of trips and mode shares by distance

(f) PlanomatX: simulated cumulated mode shares by distance

(g) Schedule recycling: simulated number of trips and mode shares by distance

(h) Schedule recycling: simulated cumulated mode shares by distance

Figure 5.10: Test VIII: Observed and simulated traffic counts (for the ease of illustration, only observed traffic counts and schedule recycling counts shown)

(a) Total of 115 counting stations within 10km circle around Zurich city centre

(b) Counting station Zurich Hardstrasse, direction NE towards Bucheggplatz (code 106311)

Figure 5.9 emphasises the number of trips and mode shares by distance. In line with the lower average distances, one can clearly see that the simulation produces slightly more short trips than reported by Microcensus. The bump at 0.2-0.5 km distance is higher, the one at 20-50 km lower than reported, and the overall peak is still at 2-5 km. The trips’ curve, in summary, looks substantially better than PlanomatX’s curve. Looking at mode shares by distance, walk mode is very well matched. Bike develops a peak in the class of 1-2 km, while prevailing in the Microcensus class of 0.5-1 km. Nevertheless, remember that simulated average bike distance is still below reported. The reason emerges in figure 5.9, where bike has a certain (unlikely) share in the 50-100 km class. Car mode’s share is slightly too high in short-distance classes, and slightly too small in the medium-distance classes, but rises as it approaches long-distance classes. In turn, public transport loses share in long-distance classes where it should increase. The error can be
overlooked, since affected trip numbers are small. The error can be traced to, first, flat public transport routing that may not be able to capture long-distance travel times appropriately (see MATSim’s router module in section 3.2.4.2) and, second, missing car traffic outside Zurich that favours the car mode. Chart 5.9h confirms that overall mode shares are best matched by schedule recycling.

Figure 5.10 compares simulated and observed car traffic counts. Both charts show that traffic counts are accurately reproduced. Major differences exist only in early morning and evening hours: obviously, due to facility opening hours. Work, as the earliest activity type, opens at 8 am, leisure as the latest closes at 8 pm. Independent from that, difference in evening counts seems unavoidable if the average home return time of 17:56h is to be met. Originating from different data sources, Microcensus and traffic counts may not be completely consistent.

5.3.2.4 Runtime

Overall runtime of the schedule recycling test was 44:13 hours. Each MATSim iteration took about 52:54 min, divided into 6:30 min of traffic flow simulation and 43:24 min of replanning runtime (0.151 sec per agent), a massive runtime reduction compared to PlanomatX. Overall runtime reduced by factor 3.9, and replanning runtime reduced by factor 4.5. Moreover, runtime became more proportional with the initial base test’s runtime; the additional activity chain dimension now leads to a replanning runtime increase by factor 10 (factor 9 on chessboard scenario), as opposed to factor 45 for PlanomatX.

Figure 5.12 breaks down a representative schedule recycling replanning period runtime consumption. PlanomatX optimization slots take about 7.4 min (2.0 + 1.0 + 4.4 min). Slightly more than 320 agents are individually optimized by PlanomatX. The distance metric slot is an average calculation. Slightly more than 5 min must be passed on to each “normal” iteration for detection of best distance metric, occurring every 10 iterations. Remaining time is taken by the assignment module handling more than 17,000 agents.

5.3.2.5 Summary

Test VIII confirms that schedule recycling generally works on real scenarios like the large-scale greater Zurich scenario with more than 170,000 agents. Schedule recycling considerably lowers simulation runtime compared to PlanomatX. The test further shows that calibrated utility function can generally reproduce real activity and travel characteristics. A major exception is still specific activity chain distribution. Differences can also be seen between PlanomatX and

---

9Remember that the scenario considers only agents crossing Zurich.
Figure 5.11: Test VIII: Comparison of runtime shares per MATSim iteration of PIx/TMC alternative and of PIx/PI alternative

![Diagram showing runtime per MATSim iteration](image)

Figure 5.12: Test VIII: Breakdown of runtime consumption of a schedule recycling replanning period

<table>
<thead>
<tr>
<th>Number of agents</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>17,375</td>
<td>43.4</td>
</tr>
<tr>
<td>16,552</td>
<td>29.5</td>
</tr>
<tr>
<td>219</td>
<td>4.4</td>
</tr>
</tbody>
</table>

schedule recycling. Schedule recycling cannot reduce the average trip travel distance to the extent PlanomatX does, and, independent from the fit with reported data, schedule recycling does not fully replicate PlanomatX’s activity chain distribution.
5.4 Discussion

Schedule recycling is a new MATSim replanning strategy. It replicates PlanomatX’s optimization functionality, but avoids optimizing each agent individually. Instead, it re-uses schedules of some individually optimized agents for other non-optimized agents. Therefore, schedule recycling’s runtime is about four times faster than PlanomatX’s. However, schedule recycling leads to different optimization results than PlanomatX:

- **Activity and trip characteristics**: On the chessboard scenario, schedule recycling leads to slightly longer activity chains (average length of 5.41 activities vs. 5.30 activities for PlanomatX). The most frequent activity chains are identical but their numbers differ. Schedule recycling optimizes toward a balanced distribution of chains, PlanomatX emphasises a few chains. Given the higher number of activities per schedule, schedule recycling can achieve a lower average trip travel distance. Average trip travel time is roughly equal to PlanomatX. On the greater Zurich scenario, the situation is reversed. PlanomatX leads to longer activity chains (also featuring a higher chain diversity), because it lowers trip travel distances more rigorously than schedule recycling. The effect is probably because location choice for re-used schedules is less efficient than for individually optimized schedules.

- **Utility**: On the chessboard scenario, schedule recycling and PlanomatX achieve the same utility level. This is not the case for the greater Zurich scenario. Here, schedule recycling falls about 7% short of PlanomatX. An obvious reason is PlanomatX’s better performance in the average trip travel distance.

Schedule recycling’s differences from PlanomatX are acceptable, given its massive runtime improvement. Nevertheless, making schedule recycling and PlanomatX more congruent presents great opportunities for further research (see chapter 7). Apart from that, MATSim’s new utility function has been calibrated for the application of schedule recycling (see chapter 6). Test VIII has shown that realistic activity and travel characteristics can be reproduced.

---

10 Measured against an initial utility score of 2.28 points at iteration 0.
Chapter 6

Specification and empirical estimation of the utility function

A new utility function for activities and travel performance has been specified, empirically estimated and manually calibrated. The utility function for the activity performance copes with a flexible number of activities in a schedule, and features an asymmetric S-shaped curve with an inflection point as presented by Joh (2004). The utility function for travel performance accounts for travel time and cost. Function parameters have been empirically estimated from Swiss Microcensus data (Swiss Federal Statistical Office, 2006) and calibrated to match reported and observed real-life data.

6.1 Specification of the new utility function

6.1.1 Problem formulation

Three sub-problems must be solved to specify a new utility function:

Defining an appropriate utility function form for activity performance
MATSim’s existing utility function features a log form for activity performance. In combination with PlanomatX and schedule recycling, the log form is problematic, resulting in unrealistic effects. When the number of scheduled activities is a dimension of the learning process, the log form leads to numerous very short activities due to its decreasing marginal utility. In other words, a schedule of two 30-minute activities of a certain type is always better than a single 60-minute activity of the same type (see example in figure 6.1). Therefore, we need a utility function capable of coping with a flexible number of schedule activities.
Figure 6.1: Dilemma of MATSim’s existing utility function: two short activities better than one long activity

Adding travel cost to utility function for travel performance
MATSim’s existing utility function for travel performance measures the utility (or disutility) of travel based on travel time. However, anyone’s experience can confirm that travel cost also impacts travel utility; thus, utility function for travel performance should also incorporate travel cost.

Including travellers’ socio-economic attributes
Finally, it is obvious that each traveller has different taste and preferences and would assign different utility scores to an identical activity/trip. The better this variation can be captured in the deterministic part of the utility function (see equation 6.7), the more accurate it is. Some taste preferences depend on socio-economic attributes, e.g., women may assess activities differently from men, or travellers without driver’s licenses may favour public transport. Utility function for both activity performance and travel performance should therefore be tested to determine what socio-demographic attributes add to their explanatory power.

6.1.2 Solution approach

The utility function structure is adapted from MATSim’s existing utility function. However, the late arrival term is dropped\(^1\)(compare with section 2.3.1.2):

\[
U_{\text{total},i} = \sum_{j=1}^{n} U_{\text{act},ij} + \sum_{j=1}^{n} U_{\text{travel},ij} 
\]  

\(^1\)The term is dropped because, first, we feel that the constraint is artificial and, second, no empirical data is available on late arrival.
where $U_{\text{total},i}$ is the total utility of agent’s $i$ schedule, $n$ the number of activities and trips, $U_{\text{act},ij}$ the (positive) utility gained from performing activity $j$, and $U_{\text{travel},ij}$ the (negative) utility gained from travelling trip $j$. However, we have modified the individual utility terms quite significantly:

- The utility function $U_{\text{act},ij}$ for the performance of activities is now

$$U_{\text{act},ij} = U_{\text{acttype},ij} \cdot S_{ij}$$

(6.2)

where $U_{\text{acttype},ij}$ is activity’s $j$ type-specific utility function (e.g., “home”) and $S_{ij}$ agent’s $i$ socio-demographic utility factor. $U_{\text{acttype},ij}$ is defined as

$$U_{\text{acttype},ij} = U_{\text{min}}^j + \frac{U_{\text{max}}^j - U_{\text{min}}^j}{(1 + \gamma_j \cdot \exp[\beta_j(\alpha_j - \text{duration}_{ij})])}$$

(6.3)

The function is an asymmetric S-shaped curve with an inflection point (see figure 6.2), originally developed in biological science and first presented in transport research by Joh (2004). It formulates an optimal activity duration by its functional form. Assuming an average value of time, the utility function features segments where the value of time is below average, and segments where it is above. Optimal activity duration is found in the latter segments. $U_{\text{min}}^j$ is the time-independent minimum utility of performing activity $j$, and $U_{\text{max}}^j$ the time-independent maximum utility of performing activity $j$. $\text{duration}_{ij}$ is the duration for which agent $i$ performs activity $j$. $\alpha_j$, $\beta_j$, and $\gamma_j$ are parameters that influence the shape of the curve:

- $\alpha_j$ indicates at what duration the function reaches its inflection point, and thus its maximum utility $U_{\text{max}}^j$. A low $\alpha_j$ shifts the curve “to the left”, a high $\alpha_j$ “to the right”.
- $\beta_j$ influences the slope of the function. A low $\beta_{ij}$ smooths the curve, a high $\beta_{ij}$ steepens it sharply.
- $\gamma_j$ determines the relative vertical position of the inflection point (see Joh 2004). A value of $\gamma_j = 1$ stands for a symmetric curve where the inflection point is halfway between minimum and maximum utility. $\gamma_j < 1$ moves the inflection point toward the minimum utility, $\gamma_j > 1$ toward the maximum utility.

$S_{ij}$ is an agent-specific multiplier to the purely activity-type-specific utility function $U_{\text{acttype},ij}$. The multiplier reflects agent’s $i$ socio-demographic characteristics impacting his utility experience. It is defined as

$$S_{ij} = (1 + \sum_m \beta_{jm} \cdot \text{attribute}_{ijm})$$

(6.4)
Figure 6.2: Generic illustration of new utility function for activity performance

\[
\text{Utility} \quad U_{\text{perf,ij}} \\
\text{Implicit minimum duration of activity} \\
\text{Implicit maximum duration of activity} \\
\text{Average value of time}
\]

\[U_{\text{perf,ij}} = \text{Average value of time} \times (1 + \sum \beta_m \cdot \text{attribute}_{ijm}) \]  

where \(\text{attribute}_{im}\) is the value of agent’s \(i\) socio-demographic attribute \(m\) (income, age, gender, etc.) and \(\beta_m\) the corresponding weight parameter.

- The travel (dis-)utility \(U_{\text{travel,ij}}\) is defined as

\[
U_{\text{travel,ij}} = \beta_{\text{time,mode},j} \cdot \text{time}_{ij} \cdot S_{ij} + 
\beta_{\text{cost,mode},j} \cdot \text{cost}_{ij} \cdot \left(\frac{\text{income}_i}{\text{averageIncome}}\right)^{\lambda_{\text{income},j}} + \text{constant}_{\text{mode},j}
\]  

\[(6.5)\]

where \(\text{time}_{ij}\) is travel time of trip \(j\) of agent \(i\), \(\text{cost}_{ij}\) travel cost, and \(\beta_{\text{time,mode},j}\) and \(\beta_{\text{cost,mode},j}\) corresponding mode-specific weight parameters. \(\left(\frac{\text{income}_i}{\text{averageIncome}}\right)^{\lambda_{\text{income},j}}\) reflects the impact of a traveller’s income level on his perception of travel cost. \(\text{income}_i\) is the traveller’s monthly income. \(\text{averageIncome}\) is the average monthly income across the population. The term allows for a continuous interaction of attributes, an alternative to inclusion of an income attribute, segmented into income classes with class-specific parameters (see [Mackie et al., 2003], [Hess et al., 2008]). A positive \(\lambda_{\text{income},j}\) means that the traveller’s sensitivity to travel cost decreases as his income increases. A negative \(\lambda_{\text{income},j}\) triggers the opposite effect (see also section 6.2.5.2, paragraph 4). \(\text{constant}_{j}\) is the mode-specific utility constant. \(S_i\) agent’s \(i\) socio-demographic utility factor affecting only travel time. As observed with the utility function for activity performance, \(S_i\) is again

\[
S_{ij} = (1 + \sum_m \beta_{jm} \cdot \text{attribute}_{ijm})
\]  

\[(6.6)\]
### 6.2 Empirical estimation of the parameters of the utility function

#### 6.2.1 Problem formulation

Aiming for a comprehensive simulation model, weight parameters of the specified utility function need to be set so that the utility function, or the simulation model, reproduces real travel behaviour. For MATSim’s existing utility function, Charypar and Nagel (2005) drew parameter values from the Vickrey scenario (see e.g., Arnott et al., 1993), representing a “reasonable but arbitrary” calibration (Charypar and Nagel, 2005). For the new utility function, an empirical estimation of parameters seems necessary, particularly with respect to the function’s increased complexity and number of parameters.

#### 6.2.2 Estimation methodology

We have followed a four-step approach to estimate MATSim’s new utility function parameters (see figure 6.3). First, relevant survey data is collected and the choice set generated (step 1). Missing survey data is added using the results of a MATSim simulation run (step 2). Estimation of utility function parameters is the core step and happens in step 3. Finally, estimation results serve as input for a MATSim simulation run and are calibrated (manually) with traffic counts. The following sections illustrate the methodology in more detail.
6.2.3 Processing survey data and generating choice set (step 1)

The estimation is based on revealed behaviour data from the Swiss Microcensus 2005 (national travel survey, Swiss Federal Statistical Office, 2006). The Microcensus comprises roughly 19,000 travellers for Switzerland overall. 4,372 travellers are included in the Greater Zurich area. For estimation of the parameters, every traveller requires a choice set comprising the chosen schedule and several non-chosen alternative schedules:

- **Chosen schedule**: The chosen schedule is obviously the schedule reported in the Microcensus. The Microcensus data set holds information on what activities each traveller performs, in which sequence, for how long, where, and which mode he uses to travel between the activities.

- **Non-chosen alternative schedules**: The non-chosen alternative schedules can be any schedules different from the traveller’s chosen one. Nevertheless, alternative schedules should reflect realistic activity and travel options and the parameter and attribute vectors $\beta_i$ and $x_i$ of each alternative $i$ must be consistent for all travellers (see model formulation, section 6.2.5). Using the assumption that we want to estimate parameters for performance of activities and travel, we must ensure that an alternative $i$ includes the same activity agenda and the same travel modes across the board for all travellers. It is then logical to assign a traveller the chosen schedule structures (activity types and modes) of other travellers as his non-chosen alternative schedule structures, s.t. those schedule structures are different from the one chosen. The scenario’s 4,372 travellers feature 976 different schedule structures, i.e. a traveller may have up to 975 alternative schedules. However, neither an estimation model requires 975 alternatives per observation, nor this is runtime efficient (Train, 1986). We have set the number of alternative schedules to nineteen\(^2\). They are chosen randomly from the set of 975 alternative schedules. For the nineteen alternative schedules, the activity locations must be adjusted:
  - The locations of all home activities are set to the traveller’s home location.
  - Primary non-home activities are assigned the traveller’s given primary activity locations as well.
  - The location choice module of section 3.2.4.1 is applied for all secondary activities.

Our choice set for estimation of parameters thus comprises 4,372 travellers. Every traveller holds one chosen schedule and 19 alternative schedules. Activity types and locations, as well as travel modes, have been set. Step 2 will complete the schedules, adding travel routes and activity/travel timings.

---

\(^2\) 1 chosen schedule + 19 alternative schedules = 20 schedules per traveller
6.2.4 Adding travel routes and adjusting travel/activity times (step 2)

Both chosen and non-chosen alternative schedules still lack travel routes. They are added using MATSim’s standard routing module (see section 2.3.1.1). Along with travel routes, travel times and effective activity durations are added. For the chosen schedules, reported travel times and activity durations need to be overwritten by the simulated ones, if they differ, to keep the estimation data consistent.

We assume traffic loads identical to those in MATSim’s greater Zurich scenario initial demand schedules as input to the routing module. Columns I) and II) of figure 6.4 show that the chosen schedules’ simulated average trip distances are about 40% higher than those reported. This is not important, since Microcensus reports crow-fly distances while MATSim outputs travelled distances. However, simulated average trip travel times are about 10% (walk) to 50% (car) lower than those reported. Since we need to overwrite reported travel times with simulated times, the data’s validity is questionable; people have chosen their reported schedules under reported travel time conditions. It is not clear whether they would have made the same decisions under substantially different simulated travel time conditions. Therefore, simulated travel times must be fitted to reported ones. Mode-specific levers have been pulled to accomplish this:

- Car mode: “Reduce MATSim’s network free-flow speeds by 30%”

Two possible reasons exist for the difference in car mode travel times:

1. MATSim’s initial demand schedules do not produce sufficient traffic to meet real traffic loads. As a consequence, simulated travel times based on those low traffic loads might be too short. However, this is not the case. Tests with traffic loads far exceeding real traffic loads still result in low average travel times. In other words, missing traffic congestion is not responsible for the discrepancy in travel times.

2. Thus, MATSim’s free-flow network speeds exceed real speeds and, as a consequence, traffic is too fast and travel times too short. This is somewhat understandable since MATSim’s free-flow network speeds reflect (legal/physical) maximum speeds, for instance 120km/h on highways. Travellers, however, do not always travel at maximum speeds; they may drive more slowly, even under free-flow conditions. We have reduced MATSim’s free-flow network speeds by 50% and 30% (see columns III) and IV) of figure 6.4 and analyzed the resulting travel times again. A reduction of 50% leads to similar reported/simulated average car trip travel times (1545sec vs. 1377sec), but we feel that this reduction is too extreme. MATSim’s street network is not as tightly-meshed as a real street network and thus, it often

---

3 The effective activity duration is the time elapsed from the arrival of the previous trip until the departure of the following trip.
Figure 6.4: Comparison of average trip distances and travel times between Swiss Microcensus schedules (as reported and as simulated in MATSim with different network free-flow speeds) and MATSim’s initial demand schedules in the greater Zurich scenario

<table>
<thead>
<tr>
<th>Trip distance (km)</th>
<th>Trip travel time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of trips per schedule</td>
<td>Car</td>
</tr>
<tr>
<td></td>
<td>1.82</td>
</tr>
</tbody>
</table>


misses the “first and last mile” of the trips, creating a reduction of travel speeds by 30%, which seems justified. The 30% reduction leads to an average car trip travel time of 1003 sec, against 1545 sec reported in the Microcensus.

- PT mode: “Set PT factor from 2.0 to 1.8”

MATSim calculates no PT routes. It determines only PT travel times assumed to be car free-flow travel times multiplied by a certain factor, that used to be 2.0 under full network speeds (see section 2.3.1.1). Given reduced network speeds, we have decreased the factor to 1.8. This is arbitrary, but lets the reported/simulated PT travel times ratio (2303 sec vs. 3169 sec) adapt a level similar to car mode (see figure 6.4).

- Bike and walk modes: “Set bike and walk speeds to 15.0 km/h and 4.0 km/h”

MATSim calculates neither bike nor walk routes. Only travel times are determined through dividing an assumed travel distance (crow-fly distance times 1.5) by an assumed speed of 15.0 km/h for bikes and 4.0 km/h for walks (see section 2.5.1.1).
travel speed. Travel speeds used to be set to 22.5km/h for bike and 4.5km/h for walk. We have reduced them to 15km/h for bike and 4km/h for walk. They now better fit the reported travel times\footnote{Note that, for bike and walk, no “missing first and last trip mile” phenomenon needs to be taken into account, since no network route calculations are involved.}

6.2.5 Estimating function parameters through an enhanced MNL model (step 3)

The parameter estimation is conducted using an enhanced Multinomial Model (MNL). The MNL model was first proposed by \cite{McFadden1974}. It assumes that the utility of an alternative \(i\) can be expressed as

\[
U_i = V_i + \varepsilon_i \tag{6.7}
\]

where \(V_i\) is the deterministic utility component and \(\varepsilon_i\) a stochastic error term. \(V_i\) is calculated as \(V_i = f(\beta_i, x_i)\). \(\beta_i\) is the vector of (taste) parameters (the parameters to be estimated) and \(x_i\) the vector of the attributes of alternative \(i\). The error term \(\varepsilon_i\) is assumed identically and independently (i.i.d.) Gumbel distributed. The choice probability of an alternative \(i\) from a given choice set \(C\) is then defined as:

\[
P(i|C) = \frac{e^{\mu V_i}}{\sum_j e^{\mu V_j}} \tag{6.8}
\]

\(\mu\) is related to the standard deviation of the Gumbel variable \((\mu^2 = \frac{\pi^2}{6})\), where, in the absence of a heterogeneous population, \(\mu\) is generally constrained to a value of 1 \cite{Schuessler2007}.

The MNL model is popular due to its uncomplicated parameter estimation (see e.g., \cite{Ben-Akiva1985}), but has the disadvantage of the Independence from Irrelevant Alternatives (IIA) property. The relative ratio of two alternatives’ choice probabilities does not depend on the existence or the characteristics of other choice alternatives (see e.g., \cite{Ben-Akiva1985, Train1986, Bierlaire1998} for more details and examples\footnote{Ben-Akiva and Lerman as well as Train refer to the “blue bus/red bus” example. Bierlaire shows the “path choice” example.}). We have therefore enhanced our MNL model with a similarity attribute in the systematic part of the utility function:

\[
V_{\text{total},i} = \sum_{j=1}^{n} V_{\text{act},ij} + \sum_{j=1}^{n} V_{\text{travel},ij} + \beta_{\text{dissim}} \cdot \text{dissim}_i^{(\lambda_{\text{dissim}})} \tag{6.9}
\]
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

subject to

\[
dissim_i^{(\lambda_{\text{dissim}})} = \begin{cases} 
\frac{\lambda_{\text{dissim}}^{-1}}{\ln \dissim_i} & \lambda_{\text{dissim}} \neq 0 \\
\ln \dissim_i & \lambda_{\text{dissim}} = 0 
\end{cases}
\]  

The attribute \(\dissim_i\) in equation 6.9 reflects the structural dissimilarity between schedules following the path-size logit or C-logit approach (Cascetta et al., 1996). \(\dissim_i\) assumes the role of a commonality factor. It compensates for the schedule’s utility bias associated with its structural dissimilarity from the alternative schedules not captured by the MNL model. It is calculated using an updated version of the Multidimensional Sequence Alignment Method (MDSAM) by Joh et al. (2002) and Joh (2004). The MDSAM will be extensively discussed in section 6.3. The Box-Cox transformation of equation 6.10 ensures an optimal fit of \(\dissim_i\) allowing for a non-linear contribution of \(\dissim_i\) (Box and Cox, 1964). In line with equations 6.1 to 6.6 and 6.9 the systematic part of the utility function to be estimated is:

\[
V_{\text{total},i} = \sum_{j=1}^{n} V_{\text{act},ij} + \sum_{j=1}^{n} V_{\text{travel},ij} + \beta_{\text{dissim}} \cdot \dissim_i^{(\lambda_{\text{income}})} = \\
= \sum_{j=1}^{n} \left[ \left( V_j^\text{min} + \frac{V_j^\text{max} - V_j^\text{min}}{(1 + \gamma_j \cdot \exp[\beta(\alpha - \text{duration}_{ij})])^{1/\gamma_j}} \right) \cdot (1 + \sum_{m} \beta_{jm} \cdot \text{attribute}_{ijm}) \right] + \\
+ \sum_{j=1}^{n} \left[ \beta_{\text{travelTime},j} \cdot \text{travelTime}_{ij} \cdot (1 + \sum_{m} \beta_{jm} \cdot \text{attribute}_{ijm}) \right] + \\
+ \beta_{\text{travelCost},j} \cdot \text{travelCost}_{ij} \cdot \left( \frac{\text{income}_i}{\text{averageIncome}} \right)^{\lambda_{\text{income},j}} + \text{constant}_{\text{mode},j} \right] + \\
+ \beta_{\text{dissim}} \cdot \dissim_i^{(\lambda_{\text{dissim}})} \]  

subject to the case differentiation of equation 6.10

6.2.5.1 Overall estimation results

We have used Biogeme\(^7\) version 1.8, to solve the MNL model (Bierlaire, 2003, 2009). Table 6.1 shows estimated parameter values. Figure 6.5 plots the corresponding utility function. The parameter values produce an extraordinarily good model fit of \(\rho^2 = 0.800\). All 30 estimated parameters are significant. A synthesis of parameter estimates follows:

\(^7\)Biogeme offers various solution algorithms for the maximum likelihood problem. DONLP2 turned out to be the only algorithm capable of dealing with the S-shaped utility function for activity performance.
Activity parameters

Figure 6.5 shows a plot of utility function for performance of activities. It features expected characteristics: an home activity reaching its maximum at a long duration, slightly shorter work and education activities, and shopping, leisure and inner home activities that reach a low maximum utility at short duration. Further characteristics are:

- All estimated $\alpha_j$ are significant. The t-test tends to improve by a rising $\alpha_j$. Relationships between activity types’ $\alpha_j$ seem reasonable, with higher $\alpha_j$ for “home” and “work” (=
Figure 6.5: Plot of the estimated utility function

(a) Utility function for the performance of activities.

(b) Utility function for the performance of travel.

- All $\beta_j$ were defined in an earlier estimate (see section 6.2.5.2 paragraph II) and fixed for the overall estimation. The joint estimation was not positive, with Biogeme getting stuck in a local extremum inferior to the estimate with fixed $\beta_j$.

- The $\gamma_j$ parameters were fixed at the default value of $\gamma_j = 1.0$ (i.e. symmetric function development). A meaningful estimation of $\gamma_j$ was not possible (see section 6.2.5.2 paragraph III).

- $V_j^{min}$ is, by definition, set to 0 utility points. $V_j^{min}$ indicates the (asymptotic) minimum utility for the performance of activity type $j$.

- All $V_j^{max}$ are significant at fairly high t-test values.

Travel parameters

Figure 6.5b shows a plot of the utility function for travel performance. Walk features the steep-
est slope. Car has a very large spread, depending on average speed. PT is nearly flat. More detailed parameters are:

- All $\beta_{travelTime,j}$ are negative except $\beta_{travelTime,PT}$. One interpretation: time spent on public transport opens up room for other tasks; for instance, work, recreation, social interaction, etc.

- $\beta_{travelCost,PT}$ is negative, as expected. $\beta_{travelCost,car}$ is positive. However, this comes together with a very negative $\beta_{travelTime,car}$. Hence, car drivers want to be fast, but care less about travel cost.

- All $constant_j$ look reasonable, particularly those discussed above $\beta_{travelTime,j}$ and $\beta_{travelCost,j}$.

- The $\lambda_{income,j}$ are as expected: $\lambda_{income,car}$ is positive, implying that travellers with above-average income prefer car mode. $\lambda_{income,PT}$ is negative, implying that public transport loses attractiveness among above-average earners.

**Socio-demographic parameters**

The model considers five socio-demographic parameters (see also section 6.2.5.2, paragraph V):

- Gender parameters $\beta_{female,act}$ and $\beta_{female,travel}$ propose that women earn slightly less utility from performing activities than men and gain more disutility for travelling. In summary, this suggests women travel less.

- Age parameters $\beta_{age,education}$ and $\beta_{age,work}$ suggest that older people derive less utility from education, and slightly less utility from work. Both of these results seem logical.

- The travel parameter $\beta_{license,car}$ implies that travellers with driver’s licenses are less sensitive to the (dis-)utility derived from car travel time than travellers without licenses. Car mode thus becomes more attractive, which again seems reasonable.

**Dissimilarity parameters**

The negative value of $\beta_{dissim}$ implies that schedules’ attractiveness decreases according to their dissimilarity to other schedules. This is interesting since, from a theoretical point of view, the opposite would be expected (see literature on IIA property and C-logit approach, e.g., Ben-Akiva and Lerman, 1985; Train, 1986; Cascetta et al., 1996; Bierlaire, 1998). From a pragmatic point of view, this finding implies that “common” schedules (= similar to the others) are more appealing than “uncommon” ones. In other words, people are creatures of habit, and dislike change. $\lambda_{dissim} = -0.949$ implies that the marginal disutility incurred by the dissimilarity term decreases by a rising value of $dissim_i$ (see section 6.2.5.2 paragraph VII).
6.2.5.2 Estimation steps

Several intermediate estimation steps were necessary to build up overall utility function and ensure a good model fit.

I) “Inner home” activity type parameters

A first estimate, with a limited number of parameters\(^8\) suggests that home activity type should be differentiated between “outer home” and “inner home” activities. “Outer home” activity is mandatory home activity at the beginning and the end of the day. It is considered one single activity lasting through midnight (see also section 2.3.1). “Inner home” activities are home activities where the traveller stops at home during the day. Thus, by definition, a non-home activity must occur between outer and inner home activities or between two inner home activities.

Inner home activity type must be estimated separately from outer home activity type. Estimating one joint home activity type, MATSim would under-estimate the duration of outer home activities, and over-estimate the duration of inner home activities (see figure 6.6). In line with this, model fit increases when differentiating the two home activity types (see table 6.2, left two columns).

II) Activities’ $\beta_j$ parameters

Default values of $\beta_j = 1.2$ have been chosen for the above estimations. Column 3 of table 6.2 shows results of an estimate with flexible $\beta_j$ parameters. $\beta_{\text{home}}$ and $\beta_{\text{work}}$ adapt values below

---

\(^8\)Note that, at this stage, schedules’ dissimilarities are taken into account linearly, excluding the $\lambda_{\text{dissim}}$ parameter.
Table 6.2: Comparison of estimations with and without inner home activity type, with flexible β_j, and with flexible β_j and γ_j

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimation without Inner Home Activity</th>
<th>Estimation with Flexible β_j</th>
<th>Estimation with Flexible β_j and γ_j</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>HeSt</td>
<td>Value</td>
</tr>
<tr>
<td>home</td>
<td>8.54</td>
<td>52.10</td>
<td>8.36</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>2.45</td>
<td>23.64</td>
<td>3.09</td>
</tr>
<tr>
<td>Inner-Home</td>
<td>-</td>
<td>-</td>
<td>0.801</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>-</td>
<td>-</td>
<td>1.3</td>
</tr>
<tr>
<td>work</td>
<td>2.07</td>
<td>19.79</td>
<td>3.8</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>1.26</td>
<td>22.27</td>
<td>1.54</td>
</tr>
<tr>
<td>education</td>
<td>1.82</td>
<td>15.61</td>
<td>2.01</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>1.25</td>
<td>14.26</td>
<td>1.17</td>
</tr>
<tr>
<td>leisure</td>
<td>0.501</td>
<td>6.04</td>
<td>0.856</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>1.19</td>
<td>23.71</td>
<td>1.08</td>
</tr>
<tr>
<td>shopping</td>
<td>-0.716</td>
<td>-3.16</td>
<td>-0.484</td>
</tr>
<tr>
<td>β</td>
<td>1.2</td>
<td>fixed</td>
<td>1.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>fixed</td>
<td>1.0</td>
</tr>
<tr>
<td>V_{max}</td>
<td>0.758</td>
<td>11.66</td>
<td>0.594</td>
</tr>
<tr>
<td>car</td>
<td>-2.07</td>
<td>-14</td>
<td>-2.63</td>
</tr>
<tr>
<td>β_{travelTime}</td>
<td>0.055</td>
<td>7.42</td>
<td>0.051</td>
</tr>
<tr>
<td>pt</td>
<td>-0.524</td>
<td>-1.81</td>
<td>-0.582</td>
</tr>
<tr>
<td>β_{travelTime}</td>
<td>0.354</td>
<td>6.24</td>
<td>0.344</td>
</tr>
<tr>
<td>β_{travelCost}</td>
<td>-0.123</td>
<td>-12.16</td>
<td>-0.124</td>
</tr>
<tr>
<td>bike</td>
<td>0.27</td>
<td>7.65</td>
<td>0.0733</td>
</tr>
<tr>
<td>β_{travelTime}</td>
<td>-1.58</td>
<td>-15.12</td>
<td>-1.47</td>
</tr>
<tr>
<td>walk</td>
<td>1.02</td>
<td>28.06</td>
<td>0.0811</td>
</tr>
<tr>
<td>β_{travelTime}</td>
<td>-1.97</td>
<td>-27.3</td>
<td>-1.91</td>
</tr>
<tr>
<td>β_{dissim}</td>
<td>-0.471</td>
<td>-20.7</td>
<td>-0.27</td>
</tr>
<tr>
<td>No of observations</td>
<td>4372</td>
<td></td>
<td>4372</td>
</tr>
<tr>
<td>L(0)</td>
<td>-13097.3</td>
<td></td>
<td>-13097.3</td>
</tr>
<tr>
<td>L(0)</td>
<td>-4201</td>
<td></td>
<td>-4097.29</td>
</tr>
<tr>
<td>ρ2</td>
<td>0.674</td>
<td></td>
<td>0.688</td>
</tr>
</tbody>
</table>

1.0, resulting in rather flat curves. β_{leisure} and β_{shopping} adapt high values, leading to steep curves for these activity types. The interpretation is that, having in mind their rather low α_j parameters, the utility of these activities is gained merely from performance, not from long durations. Fit of the model with flexible β_j (ρ^2 = 0.744) is substantially higher than that with fixed β_j parameters (ρ^2 = 0.688).
III) Activities’ $\gamma_j$ parameters

The right column of table 6.2 finally shows estimation results of a model in which both $\beta_j$ and $\gamma_j$ are flexible. The model fit ($\rho^2 = 0.754$) slightly increases compared to the model with flexible $\beta_j$ only ($\rho^2 = 0.744$). However, many parameter estimates are not significant, in particular four of six $\gamma_j$ estimates. Moreover, the remaining two parameters $\beta_{\text{home}}$ and $\beta_{\text{work}}$ produce suspiciously strong t-tests ($2.8E+05$ and $2.5E+06$). Having had some initial issues estimating the model with the flexible $\beta_j$ parameters, the estimation with flexible $\gamma_j$ parameters is no longer reliable. So, despite slightly higher model fit, all $\gamma_j$ parameters were kept at the default value of 1.0 (i.e. symmetric utility function development).

IV) Gender parameters

Left and middle columns of table 6.3 compare estimations with aggregate gender parameters $\beta_{\text{female,act}}$ and $\beta_{\text{female,travel}}$ and with disaggregate gender parameters $\beta_{\text{female,j}}$. Some of the latter disaggregate parameters look very reasonable:

- $\beta_{\text{female,work}}$ is negative implying that women derive less utility from work. This is reasonable, since women work less often than men. $\beta_{\text{female,shopping}}$ is positive, implying that women go shopping more often.

- $\beta_{\text{female,car}}$ is positive, suggesting that women incur more disutility from driving a car than men. $\beta_{\text{female,pt}}$ is negative, reflecting the attractiveness of public transport for women, who use it more often than men.

The fits of the two models are about the same ($\rho^2 = 0.746$ and $\rho^2 = 0.750$). It seemed reasonable, for simplicity, to continue with the aggregate gender parameters model that produced better results in the following estimation steps than the disaggregate gender parameters model.

V) Further socio-demographic parameters

A series of further socio-demographic parameters were tested for their explanatory power. Table 6.4 summarizes the results. Models investigating relevance of car availability, season ticket ownership, age (in discrete classes), and home municipality are either not solvable within reasonable run time (48h), or produce no significant parameters for the attributes. License ownership, age (continuous), and income are significant attributes.

---

9 The estimation is also unreliable when the $\beta_j$ parameters are kept fixed at previously estimated values and only the $\gamma_j$ parameters are estimated.
10 The tests are based on an earlier model formulation without dissimilarity attribute, which should not affect their general validity.
11 Swiss Microcensus splits municipality classes by main centres, middle centres with railroad access, middle centres without railroad access, agglomeration centres, rural areas [Swiss Federal Statistical Office 2006].
Table 6.3: Comparison of estimations with aggregate and disaggregate gender parameters as well as with further socio-demographic parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimation with aggregate gender parameters</th>
<th>Estimation with disaggregate gender parameters</th>
<th>Estimation with disaggregate age, income, Education parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>t-test</td>
<td>Value</td>
</tr>
<tr>
<td>home</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>4.38</td>
<td></td>
<td>4.68</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.249</td>
<td>10.63</td>
<td>0.249</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>3.37</td>
<td>7.27</td>
<td>10.1</td>
</tr>
<tr>
<td>incomeHome</td>
<td>0.352</td>
<td>8.65</td>
<td>0.345</td>
</tr>
<tr>
<td>$\beta$</td>
<td>15.2</td>
<td>4.26</td>
<td>15.2</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>1.51</td>
<td>22.04</td>
<td>1.58</td>
</tr>
<tr>
<td>work</td>
<td>5.61</td>
<td>30.47</td>
<td>5.77</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.491</td>
<td>22.75</td>
<td>0.491</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>3.71</td>
<td>19.67</td>
<td>3.39</td>
</tr>
<tr>
<td>education</td>
<td>2.07</td>
<td>10.48</td>
<td>2.05</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2.23</td>
<td>5.29</td>
<td>2.23</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>1.35</td>
<td>15.76</td>
<td>2.06</td>
</tr>
<tr>
<td>leisure</td>
<td>0.0585</td>
<td>8.05</td>
<td>0.0585</td>
</tr>
<tr>
<td>$\beta$</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>1.36</td>
<td>26.63</td>
<td>1.33</td>
</tr>
<tr>
<td>shopping</td>
<td>0.0624</td>
<td>6.39</td>
<td>0.0637</td>
</tr>
<tr>
<td>$\beta$</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>$\gamma_{\text{max}}$</td>
<td>1.33</td>
<td>21.55</td>
<td>1.05</td>
</tr>
<tr>
<td>all $y$ fixed at 1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car $\beta_{\text{travelTime}}$</td>
<td>-2.56</td>
<td>-16.64</td>
<td>-2.48</td>
</tr>
<tr>
<td>$\beta_{\text{travelCost}}$</td>
<td>0.059</td>
<td>12.47</td>
<td>0.0571</td>
</tr>
<tr>
<td>pt constant</td>
<td>-0.63</td>
<td>19.77</td>
<td>-0.636</td>
</tr>
<tr>
<td>$\beta_{\text{travelTime}}$</td>
<td>0.449</td>
<td>1.78</td>
<td>0.559</td>
</tr>
<tr>
<td>$\beta_{\text{travelCost}}$</td>
<td>-0.109</td>
<td>5.33</td>
<td>-0.116</td>
</tr>
<tr>
<td>bike constant</td>
<td>0.3001</td>
<td>0.75</td>
<td>0.0474</td>
</tr>
<tr>
<td>$\beta_{\text{travelTime}}$</td>
<td>-1.15</td>
<td>-11.46</td>
<td>-0.352</td>
</tr>
<tr>
<td>walk constant</td>
<td>0.758</td>
<td>13.37</td>
<td>0.755</td>
</tr>
<tr>
<td>$\beta_{\text{travelTime}}$</td>
<td>-1.48</td>
<td>-13.53</td>
<td>-1.53</td>
</tr>
<tr>
<td>$\beta_{\text{female act}}$</td>
<td>0.0792</td>
<td>2.04</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{female travel}}$</td>
<td>0.153</td>
<td>5.32</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{female home}}$</td>
<td>-</td>
<td>-</td>
<td>0.162</td>
</tr>
<tr>
<td>$\beta_{\text{female innerHome}}$</td>
<td>-</td>
<td>-</td>
<td>-0.0226</td>
</tr>
<tr>
<td>$\beta_{\text{female work}}$</td>
<td>-</td>
<td>-</td>
<td>-0.0344</td>
</tr>
<tr>
<td>$\beta_{\text{female education}}$</td>
<td>-</td>
<td>-</td>
<td>-0.0103</td>
</tr>
<tr>
<td>$\beta_{\text{female leisure}}$</td>
<td>-</td>
<td>-</td>
<td>0.0527</td>
</tr>
<tr>
<td>$\beta_{\text{female shopping}}$</td>
<td>-</td>
<td>-</td>
<td>0.556</td>
</tr>
<tr>
<td>$\beta_{\text{female car}}$</td>
<td>-</td>
<td>-</td>
<td>0.21</td>
</tr>
<tr>
<td>$\beta_{\text{female pt}}$</td>
<td>-</td>
<td>-</td>
<td>-0.114</td>
</tr>
<tr>
<td>$\beta_{\text{female bike}}$</td>
<td>-</td>
<td>-</td>
<td>0.833</td>
</tr>
<tr>
<td>$\beta_{\text{female walk}}$</td>
<td>-</td>
<td>-</td>
<td>0.00075</td>
</tr>
<tr>
<td>$\beta_{\text{sale home}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{sale innerHome}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{sale work}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{sale education}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{sale leisure}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{sale shopping}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income home}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income innerHome}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income work}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income education}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income leisure}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{income shopping}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{license car}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{license pt}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{license bike}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\beta_{\text{license walk}}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\alpha_{\text{dissim}}$</td>
<td>-0.368</td>
<td>-22.6</td>
<td>-0.37</td>
</tr>
<tr>
<td>No of observations</td>
<td>4372</td>
<td>4372</td>
<td>4372</td>
</tr>
<tr>
<td>L(0)</td>
<td>-13093.3</td>
<td></td>
<td>-13093.3</td>
</tr>
<tr>
<td>L(β)</td>
<td>-333143</td>
<td></td>
<td>-327548</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.746</td>
<td></td>
<td>0.75</td>
</tr>
</tbody>
</table>
Table 6.4: Summary of further socio-demographic parameters estimation results

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type of attribute</th>
<th>&quot;&lt;attribute&gt;_travel_mode&quot;</th>
<th>&quot;&lt;attribute&gt;_act_type&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>License ownership</td>
<td>discrete</td>
<td>✓ 3 of 4 parameters significant</td>
<td>n/a</td>
</tr>
<tr>
<td>Car availability</td>
<td>discrete</td>
<td>no, not solvable</td>
<td>n/a</td>
</tr>
<tr>
<td>Season ticket ownership</td>
<td>discrete</td>
<td>no, not solvable</td>
<td>n/a</td>
</tr>
<tr>
<td>Age</td>
<td>continuous</td>
<td>no, not significant</td>
<td>✓ 3 of 5 parameters significant</td>
</tr>
<tr>
<td>Municipality</td>
<td>continuous</td>
<td>no, not significant</td>
<td>no, not solvable</td>
</tr>
<tr>
<td>Income</td>
<td>continuous</td>
<td>no, not significant</td>
<td>✓ 3 of 5 parameters significant</td>
</tr>
</tbody>
</table>

The latter attributes have been further tested in a joint estimation (see table 6.3, right column). The joint model is difficult to solve (runtime of 2 days and 22 hours), but produces a good model fit ($\rho^2 = 0.794$). 11 of the 16 additional parameter estimates are significant. As a trade-off between model runtime and fit, the parameters showing high significance, namely $\beta_{work,age}$, $\beta_{education,age}$, and $\beta_{car,license}$, were retained.

VI) Travel cost parameters

Beyond standard travel cost formulation $\beta_{\text{travelCost},j} \cdot \text{travelCost}_{ij}$ applied in previous estimations, travel cost can also be set against travellers’ monthly household incomes:

- $\beta_{\text{travelCost},j} \cdot \text{travelCost}_{ij} / \text{income}_i$: Travel cost is divided by a traveller’s monthly household income;

- $\beta_{\text{travelCost},j} \cdot \text{travelCost}_{ij} / \ln \text{income}_i$: Travel cost is divided by the natural logarithm of the traveller’s monthly household income. This formulation assumes a decreasing marginal sensitivity by rising income.

- $\beta_{\text{travelCost},j} \cdot \text{travelCost}_{ij} \cdot \left( \frac{\text{income}_i}{\text{averageIncome}} \right)^{\lambda_{\text{income},j}}$: Travel cost is multiplied with a continuous cost-income interaction term reflecting the traveller’s cost sensitivity (depending on his income relative to the population’s average income). A positive $\lambda_{\text{income},j}$ implies that travellers’ sensitivity to travel cost decreases as their incomes rise. A negative $\lambda_{\text{income},j}$ indicates the opposite. The formulation was first presented by Mackie et al. (2003) and applied, for instance, by Hess et al. (2008). It requires no a priori assumption on cost-income interaction. It also overcomes an (arbitrary) discrete segmentation of travellers by income classes with respective $\beta$ parameters.

\[12\] Note that $\beta_{\text{pt,license}}$, however, produces a suspiciously high t-test of -2.3E+07.
Table 6.5 shows estimation results of corresponding models. Model 4, with the continuous cost-income interaction term, performs best, featuring the highest model fit ($\rho^2 = 0.79$) and preserving meaningful $\alpha_{\text{home}}$ and $\alpha_{\text{innerHome}}$ parameters. Models 1, 2 and 3 show less model fit and “mix up” $\alpha_{\text{home}}$ and $\alpha_{\text{innerHome}}$.

### VII) Dissimilarity attribute $\text{dissim}_i$

All previous models included a linear formulation of the dissimilarity attribute $\text{dissim}_i$. Box-
Figure 6.7: Comparison of linear dissimilarity term and box cox transformation

Cox transformation offers a possibility to test for non-linear relationships (Box and Cox, 1964), and it is the last estimation step. Results are shown in the overall results table 6.1. The model fit again improves from $\rho^2 = 0.79$ to $\rho^2 = 0.80$. Figure 6.7 illustrates the impact of dissimilarity parameters. $\beta_{\text{dissim}}$ leads to a strong, general reduction of utility. $\lambda_{\text{dissim}}$ ensures a diminishing marginal utility about a rising $\text{dissim}_i$ (see figure 6.7 upper chart). Plotting schedules’ realized values of $\text{dissim}_i$ against marginal utility, one observes that most realized values of $\text{dissim}_i$ are located in the area where the marginal utility of the Box-Cox transformed curve flattens out (see figure 6.7 upper chart). Thus, “all” schedules are diminished by about 120 utility points, while relative impact is much lower. Importantly, the core message remains untouched; schedules’ attractiveness decreases with their dissimilarity to other schedules.

\footnote{See, for instance Weis (2006), for a sensitivity discussion of $\lambda_{\text{dissim}}$.}
6.2.6 Calibrating function parameters with Microcensus and traffic counts (step 4)

6.2.6.1 Differences between reported/observed and simulated behaviour

Despite the good model fit, estimated utility function does not reproduce reality. Simulation results differ substantially from reported and observed data (Microcensus, traffic counts) when running MATSim\textsuperscript{14} with estimated utility function (see figure 6.8):

- **Activities:** Estimated utility function leads to an average activity chain length of 11.53 activities\textsuperscript{15}. Microcensus 2005 reports an average length of only 4.65 activities. Most obvious is the difference for “inner home” and “shopping” activity types (see figure 6.8a, lower chart). Inner home would be performed 2.56 times a day, while Microcensus reports 0.50 inner home activities per day. Shopping would be performed 3.76 times a day, while Microcensus reports 0.37 shopping activities per day. In line with the strongly increased daily number of activities, MATSim’s average home departure/return times are before/after the reported ones and average activity durations are too low (see figure 6.8a, upper chart). Altogether, less time is spent on performing activities than reported by Microcensus.

- **Travel:** Total number\textsuperscript{16} of trips per schedule is 10.51 for the estimated utility function. Microcensus 2005 reports only 3.65 trips. Estimated utility function, more than other factors, leads to an over-use of car mode (7.28 vs. 1.83 trips). Bike is also used too often, PT too little. Trip travel times are underestimated throughout, but sum to a schedule total that is more than twice the reported one. Mode trip distances are either too high or too low. Their schedule total is again significantly too high. Finally, it is logical that simulated car traffic counts are substantially too high in comparison to observed counts (see figure 6.9).

\textsuperscript{14}MATSim was run for 20 iterations, sufficient to see the substantial differences between simulation outcome and reality. Schedule recycling was used as replanning strategy.

\textsuperscript{15}Note that daily “home” frequency is 1. However, home must be considered either once or twice within the activity chain length, depending on whether it is a single home activity (activity chain “home”), or whether other activities happen between (activity chain “home-non-home-activity>home”). Accordingly, individual activity type frequencies do not add up to the total number of activities per schedule.

\textsuperscript{16}Total number of trips per schedule does not correspond with total number of activities per schedule due to “home” phenomenon. See previous footnote.
6.2.6.2 Root causes of differences between reported/observed data and simulation results

Differences between reported/observed behaviour and simulation results are unfortunate. Root causes must be sought in model structure and model parameters\(^\text{[17]}\). Sevcikova et al. (2007) state five general sources of uncertainty for stochastic (transport) models: measurement errors, systematic errors, uncertainty about model structure, uncertainty about model input parameters, and
Figure 6.9: Observed and simulated traffic counts for estimated utility function

(a) Total of 115 counting stations within 10km circle around Zurich city centre

(b) Counting station Zurich Hardtstrasse, direction NE towards Bucheggplatz (code 106311)

- **Model structure:**

  - **Structure of the choice set:** Our choice set alternatives represent different schedule structures (see section 6.2.3). A schedule structure is an agenda of activity types and mode choices but it does not take their order into account. For instance, the choice set does not differentiate activity-mode-chains “home-pt-work-walk-shopping-pt-home” and “home-walk-shopping-pt-work-pt-home”. The missing ordering information seems to lead to a positive discrimination of short activity types, i.e. “inner stochasticity. Measurement errors, systematic errors and stochasticity are unlikely to be responsible: Some unexplained inconsistencies between traffic count data and Microcensus data exist (see discrepancy in home return time in section 5.3.2.3). However, this could only explain differences between traffic counts and simulation results, not between Microcensus data and simulation results, since both model validation and estimation are based on Microcensus data. Systematic errors occur between PlanomatX and schedule recycling, since schedule recycling leads to systematically different results from PlanomatX. However, both generally improve schedule utility, implying that utility function parameters are mis-set. Finally, impact of stochasticity has been tested on the chessboard scenario and was negligible.
Therefore, longer activity types, such as work and education, are shortened, and many shorter activity types inserted. In line with this, the ratio of same consecutive activity types in an activity chain (e.g., “...-shop-shop-...”) rises from 0.12 as reported in Microcensus to 2.05 in simulated results.

- **Dissimilarity attribute:** Our MNL model is enhanced by a dissimilarity attribute. According to the estimation, this attribute ensures that a schedule’s attractiveness decreases with its dissimilarity to the other schedules, and it reduces, most of all, the attractiveness of long schedules. However, in the simulation, the dissimilarity attribute has no direct effect anymore since it is not part of the operational utility function. Thus, long schedules are no longer afflicted with the disadvantage incurred by the dissimilarity attribute and become more attractive.

- **Model parameters:**

  - **Monetary cost of activity performance:** We have omitted parameters explaining monetary cost/benefit of activity performance because of missing input data availability. Inclusion of such parameters would add explanatory power to the utility function, since e.g., shopping attractiveness would not only depend on time availability, but also monetary solvency (c.p.). Current overestimation of daily shopping frequency would be likely to fade.

### 6.2.6.3 Calibration methodology

All root causes above require substantial modelling and time efforts to be solved, well beyond the scope of this Ph.D project (but clearly calling for enhancements in the future): the choice set structure should include ordering information, particularly parameters indicating activity type subsequence, daily number of activity type occurrences, and/or overall time spent per activity type. Dissimilarity attribute effects are best avoided by choosing a more advanced discrete choice model (e.g., nested logit models), making the dissimilarity attribute entirely superfluous. Finally, inclusion of monetary cost/benefit of activity performance will require innovative ideas in data collection and processing.

Instead, a manual calibration of the estimated utility function parameters is done. Seeking to harmonize simulated with observed/reported data, we focus on manipulation of the most

---

18 We applied quite rigid opening hours of 18:00h to 21:00h to the “leisure” activity type, meaning leisure cannot benefit the same way as inner home and shopping.

19 Microcensus’ (hence, the choice set’s) average activity chain length is 4.65. A natural lower bound is a chain length of 1 (“home”). The longest schedule reported in Microcensus features a length of 17 activities (“home-work-leisure-work-shop-home-leisure-home-shop-work-home-leisure-leisure-leisure-home”). It is clear that such long schedules offer more opportunity to be dissimilar to a schedule with 4.65 activities than a schedule with a chain length of one activity.

20 Note that the calibration makes an iterative approach of estimation and travel time adjustment redundant, as presented by e.g., Vrtic (2003), chapter 6, or de Palma et al. (2007). Also note that a calibration approach as
relevant parameters, particularly $V_{j}^{max}$, $\alpha_{j}$, $\beta_{j}$, $\beta_{travelTime,j}$, $constant_{j}$. The manual process is cumbersome, but has the advantage that a limited number of trial simulation runs (here about 70) leads to a reasonable solution. Its drawback is that the solution is likely to be sub-optimal. More formal approaches, for instance neural networks (Arbib 2003), may produce better calibration results but require several hundred simulation runs, given the number of parameters to be calibrated. This seems impractical to achieve within reasonable time constraints, given our simulation runtimes.

presented in Flötteröd et al. (2009); Flötteröd (2009) is not appropriate, because it can be seen as correction of the alternative-specific constants of the choice set (=schedule structure) alternatives, all being 0 in our case and not part of our utility function. Moreover, the approach would influence only traffic counts calibration, not schedule elements calibration (activity types frequencies, durations, and so on).
Table 6.6: Estimated and manually calibrated utility function parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimated parameters</th>
<th>Manually calibrated parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>t-test</td>
<td></td>
</tr>
<tr>
<td>home</td>
<td>0.32</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>0.249</td>
<td></td>
<td>12.00</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>0.429</td>
</tr>
<tr>
<td>V_min</td>
<td>6.00</td>
<td></td>
<td>1.00</td>
</tr>
<tr>
<td>V_max</td>
<td>3.59</td>
<td></td>
<td>6.41</td>
</tr>
<tr>
<td>innerHome</td>
<td>0.249</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>15.2</td>
<td></td>
<td>1.50</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>17.50</td>
</tr>
<tr>
<td>V_min</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>V_max</td>
<td>1.92</td>
<td></td>
<td>1.10</td>
</tr>
<tr>
<td>work</td>
<td>3.86</td>
<td>26.24</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>0.491</td>
<td></td>
<td>4.50</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>0.568</td>
</tr>
<tr>
<td>V_min</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>V_max</td>
<td>4.37</td>
<td></td>
<td>5.60</td>
</tr>
<tr>
<td>education</td>
<td>1.49</td>
<td>11.54</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2.29</td>
<td></td>
<td>6.60</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>2.50</td>
</tr>
<tr>
<td>V_min</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>V_max</td>
<td>4.93</td>
<td></td>
<td>5.80</td>
</tr>
<tr>
<td>leisure</td>
<td>0.0488</td>
<td>6.31</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>100.0</td>
<td></td>
<td>2.60</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>5.60</td>
</tr>
<tr>
<td>V_min</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>V_max</td>
<td>5.09</td>
<td></td>
<td>4.60</td>
</tr>
<tr>
<td>shopping</td>
<td>0.0453</td>
<td>5.22</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>100.0</td>
<td></td>
<td>0.70</td>
</tr>
<tr>
<td>γ</td>
<td>1.00</td>
<td></td>
<td>5.60</td>
</tr>
<tr>
<td>V_min</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>V_max</td>
<td>1.94</td>
<td></td>
<td>5.60</td>
</tr>
<tr>
<td>car</td>
<td>β_travelTime</td>
<td>-2.77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>β_travelCost</td>
<td>-0.0274</td>
<td>-3.10</td>
</tr>
<tr>
<td></td>
<td>λ_income</td>
<td>0.0185</td>
<td>0.85</td>
</tr>
<tr>
<td>pt</td>
<td>constant</td>
<td>-0.578</td>
<td>-0.39</td>
</tr>
<tr>
<td>β_travelTime</td>
<td>0.582</td>
<td></td>
<td>0.563</td>
</tr>
<tr>
<td>β_travelCost</td>
<td>-0.117</td>
<td></td>
<td>-0.17</td>
</tr>
<tr>
<td>λ_income</td>
<td>-0.227</td>
<td></td>
<td>0.22</td>
</tr>
<tr>
<td>bike</td>
<td>constant</td>
<td>0.165</td>
<td></td>
</tr>
<tr>
<td>β_travelTime</td>
<td>-1.07</td>
<td></td>
<td>-0.37</td>
</tr>
<tr>
<td>walk</td>
<td>constant</td>
<td>0.354</td>
<td></td>
</tr>
<tr>
<td>β_travelTime</td>
<td>-1.48</td>
<td></td>
<td>0.40</td>
</tr>
<tr>
<td>β female_act</td>
<td>-0.0777</td>
<td></td>
<td>-0.057</td>
</tr>
<tr>
<td>β female_travel</td>
<td>0.0797</td>
<td></td>
<td>0.0797</td>
</tr>
<tr>
<td>β age_education</td>
<td>-0.046</td>
<td></td>
<td>-0.016</td>
</tr>
<tr>
<td>β age_work</td>
<td>-0.00984</td>
<td></td>
<td>-0.00984</td>
</tr>
<tr>
<td>β license_car</td>
<td>-0.537</td>
<td></td>
<td>-0.25</td>
</tr>
<tr>
<td>β dissim</td>
<td>-1.39</td>
<td></td>
<td>-1.30</td>
</tr>
<tr>
<td>λ dissim</td>
<td>-0.349</td>
<td></td>
<td>-0.349</td>
</tr>
<tr>
<td>β repeat</td>
<td>-</td>
<td></td>
<td>-0.30</td>
</tr>
</tbody>
</table>

6.2.6.4 Calibration results

Utility function has been calibrated for schedule recycling, since this is likely to be applied more often than PlanomatX due to its lower simulation runtimes. Table 6.6 shows estimated
and calibrated utility function parameters, while figure 6.10 plots calibrated utility function. In summary, activities’ target durations have been lengthened (higher $\alpha$), their utility curve slopes balanced out (higher $\beta$ when original $\beta$ very low, lower $\beta$ when original $\beta$ very high), and most maximum utilities $V_{max}$ decreased. For travel parameters, car, bike, and walk modes have been made less attractive to boost public transport. In more detail, the following parameter adjustments were carried out:

- **Home**: The home activity type received a lengthened target duration (higher $\alpha$), a slightly steeper utility curve (higher $\beta$), and a lower maximum utility $V_{max}$.

- **Inner home**: Inner home activity type was also provided with a lengthened target duration (higher $\alpha$) and a slightly steeper utility curve (higher $\beta$). The maximum utility $V_{max}$ was increased.

- **Work**: Adjustments of inner home activity type have been marginal. Target duration has become slightly higher (higher $\alpha$), the utility curve slightly steeper (higher $\beta$). The maximum utility $V_{max}$ has been kept almost fixed.

- **Education**: Education activity type was given a lengthened target duration (higher $\alpha$) and a slightly steeper utility curve (higher $\beta$). The maximum utility $V_{max}$ was decreased.

- **Leisure**: Leisure activity type required a drastically increased target duration (higher $\alpha$). In turn, its utility curve has been flattened (lower $\beta$). The maximum utility $V_{max}$ remains almost fixed.

- **Shopping**: As with leisure activity type, shopping activity type has been provided with a longer target duration (higher $\alpha$). Its utility curve has been flattened (lower $\beta$). The maximum utility $V_{max}$ has fallen drastically.

To account for missing ordering information in our choice set and the resulting positive discrimination of shorter activity types, we have introduced a $\beta_{repeat}$ parameter. The parameter penalises an activity if it features the same type as its preceding activity. Utility gained from the activity is reduced by the parameter factor, here by -50%. Given these factors, the ratio of same consecutive activity types falls and, hence, the positive discrimination of shorter activity types is lessened.

The travel parameters have been adjusted as follows:

- **Car**: Disutility of car travel time has been slightly lowered (lower $\beta_{travelTime}$). In addition, the license factor (that decreases the disutility of car travel for persons with a driver’s license) $\beta_{licenseCar}$ has been reduced. That step was necessary to smooth car travel use amplitude.
• **Public transport**: The constant has been slightly increased.

• **Bike**: The constant has been slightly decreased.

• **Walk**: Walk mode required an adjustment of both constant and travel time parameters $\beta_{\text{travelTime}}$. Both parameters have been manipulated to favour a less attractive walk mode.

Further adjustments have been as follows:

• Minimum shopping duration was reduced from 60 min to 30 min to account for average shopping duration of 44 min, as reported by Microcensus.

• Facility opening hours are 7 am to 6 pm for all work types, 8 am to 4 pm for all education types, 10 am to 6 pm for shopping, and 12 pm to 8 pm for leisure.

The greater Zurich scenario tests VI and VIII were run with the utility function above. The schedule recycling test VIII particularly shows that calibrated utility function parameters reproduce reality significantly better than estimated parameters (see pages 117 et sqq.). Simulated and reported/observed data generally match well. Potential for improvement still exists in activity chains distribution. Specific occurrence numbers do not yet correspond with reported Microcensus data, particularly those that include leisure activity type. However, in summary, average frequency and durations of activity types, as well as trip numbers and mode shares, are nicely reproduced. Observed and simulated traffic counts correspond very well. See section 5.3.2 for the detailed analysis.

### 6.3 Definition of schedules’ dissimilarities

Our MNL model formulation features dissimilarity attribute $dissim_i$ in the systematic part of the utility function to overcome its Independence of Irrelevant Alternatives (IIA) property (see equation 6.9). $dissim_i$ reflects the structural dissimilarity between a person’s chosen schedule $i$ and set of alternative schedules $j$, $i \neq j$. It considers the dimensions activity chain, location and mode choice. We have adopted the Multi-Dimensional Sequence Alignment Method by Joh et al. (2002) and Joh (2004) to calculate $dissim_i$.

#### 6.3.1 Problem formulation

Several schedule dimensions consist of discrete sequences of information (for instance “home-work-home” as activity chain, “car-car” as mode choice, and so forth). Comparing those dimen-
6.3.2 Solution measure and algorithm

Our dissimilarity measure is based on the Multi-Dimensional Sequence Alignment Method by Joh et al. (2002) and Joh (2004). Sequence alignment methods, originally developed in biological science, compare sequences of discrete information, e.g., genes. They measure how many “moves” of sequence elements are necessary before the sequence matches another. Comparing activity chains, Wilson (1998) first presented sequence alignment methods in transport literature. Joh enhanced the methodology and introduced a Multi-dimensional Sequence Alignment Method integrating the comparison of several discrete schedule dimensions (activity chain, mode and location choice, accompanying person).

We have adopted Joh’s multi-dimensional similarity measure. The solution algorithm proposed is an enhanced version of Joh’s Dynamic Programming Approach, solving for $dissim_i$ in three core steps (see figure 6.11).
6.3.2.1 Defining uni-dimensional dissimilarity of discrete attributes (step 1)

Step 1 finds the uni-dimensional dissimilarities of two schedules’ discrete dimensions (i.e. activity chain, location and mode choice). It applies the sequence alignment method by Kruskal (1983), complemented by a position-sensitivity by Joh (2004): two sequences $s$ (“source”) and $g$ (“target”) are compared. They feature $m + 1$ and $n + 1$ elements ($s = [s_0, \ldots, s_m]$, $g = [g_0, \ldots, g_n]$), where $s_0$ and $g_0$ are null elements that initialize the sequences. Sequences can be made identical by three moves: “insertion”, “deletion”, and “identity”. A two-dimensional comparison table helps understand the concept (see figures 6.12a and b). The source sequence $s$ is the ordinate, the target sequence $g$ the abscissa. The table entries store the effort required to equalize $s_i$ with $g_j$. A horizontal move from $j - 1$ to $j$ is an insertion of the target element $g_j$, a vertical move from $i - 1$ to $i$ is a deletion of the source element $s_i$. A diagonal move from $(i - 1, j - 1)$ to $(i, j)$ is an identity move if the source element $s_i$ equals the target element $g_j$. All moves are associated with weights representing the moves’ efforts:

$$
w_0 = w_{\text{insertion}} = w_{\text{deletion}} = w_{\text{identity}} = \frac{|i - j|}{\max[m, n]} \cdot w_0$$  \hspace{1cm} (6.12)

where $w_0$ is a default weight value (generally $w_0 = 1$). Equation (6.12) ensures that the identity move’s effort increases by the distance between element positions in source and target sequence (“position-sensitivity”). Every table entry $(i, j)$ indicates the minimum effort\footnote{See Joh (2004) for a description of the simple algorithm to calculate minimum effort.} to change (sub-)sequence $s$ into (sub-)sequence $g$ up to positions $(i, j)$. Overall effort when $(i, j) = (m, n)$ is called Levenshtein distance. The corresponding path from $(0, 0)$ to $(m, n)$ is optimal trajectory. Several optimal trajectories can exist.
6.3.2.2 Defining multi-dimensional dis-similarity of discrete attributes (step 2)

Given the three discrete schedule dimensions - activity chain, location and mode choice - one could consider their average similarity as aggregate dis-similarity measure. However, Joh (2004) provides evidence of dimension interdependencies relevance. He therefore suggests a multi-dimensional sequence alignment method that solves for an integrated trajectory of cross-dimensional moves, not uni-dimensional moves (see figure 6.13). The move weights are extended as follows:

\[
  w_{\text{move}}^{\text{multi}} = \max_k \left[ w_{\text{move}}^{\text{uni}} \cdot \beta_k \right]; \quad k = \{AC, LC, MC\}
\]  

where \( \beta_k \) denotes the importance of the dimensions activity chain (AC), location choice (LC), and mode choice (MC). Maximum term means that cross-dimensional moves are priced by the most important (= most “expensive”) dimension involved, but not additively by the sum of all involved dimensions. Therefore, the multi-dimensional sequence alignment value is generally lower (in no case higher) than the average value of the uni-dimensional sequence alignments. Multi-dimensional formulation widens the solution space significantly, since multi-dimensional optimal trajectory(ies) may consist of both optimal and non-optimal uni-dimensional trajectories. Joh presents a genetic algorithm, a dynamic programming algorithm and a hybrid of both to solve the problem. In a nutshell, the genetic and hybrid algorithm slightly outperform the dynamic programming approach in quality of dissimilarity measurement, but are complex to implement and very runtime-consuming. We have decided to re-implement the dynamic programming approach, including some intuitive upgrades to Joh’s algorithm.

Dynamic programming approach assumes that most uni-dimensional optimal trajectories run along the diagonal area of the comparison table (Kruskal, 1983). Those trajectories involve multiple identity moves that cost less than insertion/deletion moves (see equation 6.12). For

---

Figure 6.13: Comparison of a two-dimensional sequence

Source: Joh (2004)
every dimension, Joh’s dynamic programming approach (randomly) selects one optimal uni-
dimensional trajectory close to the diagonal. If possible, it then merges the uni-dimensional
moves into common cross-dimensional moves and sums up the overall effort to equalize both
schedule sequences. Our version of the dynamic programming approach enhances the process
as it handles one dimension after the other, taking advantage of knowing the optimal trajectories
of the previous dimensions. Finding its way through the comparison table and looking for an
optimal uni-dimensional trajectory, our algorithm verifies whether a potential move has already
been part of the optimal trajectory of a previous dimension. If so, it favours the move since it
can be merged into a cross-dimensional move. This boosts the probability of cross-dimensional
moves, thus improving the quality of the dissimilarity measure. Table 6.14 illustrates com-
parison tables of two example schedules with largely overlapping AC and LC trajectories, as
calculated by our re-implemented algorithm.23

23One can clearly observe characteristics of the re-implemented algorithm: for instance, LC trajectory in-
cludes the insertion/deletion move \((s_3, g_2) \rightarrow (s_3, g_3) \rightarrow (s_4, g_3)\). The deletion move builds a cross-dimensional
move with the AC deletion move \((s_3, g_3) \rightarrow (s_4, g_3)\). The algorithm did not choose the insertion/deletion move
\((s_3, g_2) \rightarrow (s_4, g_2) \rightarrow (s_4, g_3)\) that would yield the same uni-dimensional LC cost but does not allow for the cross-
dimensional move. On the other side, the LC trajectory follows the path \((s_2, g_2) \rightarrow (s_2, g_3) \rightarrow (s_3, g_3)\). It could
share the cross-dimensional insertion move \((s_2, g_2) \rightarrow (s_3, g_3)\) if it followed \((s_2, g_2) \rightarrow (s_3, g_2) \rightarrow (s_3, g_3)\).
Bearing in mind the order AC, MC, LC, it does not do so, because the LC trajectory was unknown when MC
trajectory was to be found, and the algorithm randomly chose the “wrong” path. Finally, the algorithm determined
LC trajectory, knowing the MC moves, but could not follow that path because it was invalid for the LC dimension.
6.3.2.3 Building the aggregate similarity measure (step 3)

Step 3 finalizes the algorithm. It sets $dissim_i$ as the average of step’s 2 multi-dimensional distances of schedule $i$ with all nineteen alternative schedules $j$, $i \neq j$. The higher $dissim_i$ is, the lower schedule’s $i$ similarity with all alternative schedules $j$. Figure 6.7, upper chart, contains a histogram of $dissim_i$ for all 87440 schedules of the choice set. Note that labels of the x-axis represent the interval $[\text{label};\text{label}+1]$, for instance “6” stands for interval $[6; 7]$.

6.3.2.4 Runtime performance of algorithm

The algorithm handles our choice set’s 87,440 plans in about 27.3 sec (= 0.3 msec per schedule). Step 1 takes 17.8s and step 2 8.3s. Step 3 and overhead calculations consume 1.3 sec.

6.4 Determination of missing attributes in the greater Zurich scenario’s synthetic population

MATSim’s new utility function refers to socio-demographic attributes “monthly household income” and “season ticket ownership”, among others. These attributes were not default attributes of the greater Zurich scenario’s synthetic population (see section 2.3.1.1) and needed to be generated.

6.4.1 Income

We have adopted a three-step methodology to add agents’ income to the data set. Step 1 takes agents’ personal income (approximated by educational qualification) into account, step 2 incorporates agents’ regional income level (by municipality), and step 3 introduces some stochastics:

1. **Definition of average income by educational qualification**: Swiss Microcensus offers information on income and highest educational qualification by person (Swiss Federal Statistical Office, 2006). Determining persons’ average income by educational qualification shows a clear relationship between the two attributes (see table 6.7). Building on that relationship, we assigned every agent of the Greater Zurich scenario with an income according to his educational qualification.\(^{25}\)

---

\(^{24}\)20 plans/individual · 4372 individuals = 87440 plans.

\(^{25}\)Educational qualification is a default attribute and available for every agent.
### Table 6.7: Average income by educational qualification

<table>
<thead>
<tr>
<th>Education type</th>
<th>Microcensus code</th>
<th>MATSim code</th>
<th>Description</th>
<th>Income (CHF)</th>
<th>Weighted number of persons</th>
<th>Weighted income contribution (CHF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-8, -9</td>
<td>1</td>
<td>keine Ausbildung, noch nicht schulpflichtig</td>
<td>3024.91</td>
<td>72.40</td>
<td>46.80</td>
</tr>
<tr>
<td>2</td>
<td>11, 12</td>
<td>2</td>
<td>obligatorische Schule, Diplomhöherelehrer</td>
<td>3752.91</td>
<td>720.21</td>
<td>577.60</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>3</td>
<td>Berufsfachschule, Lehre, Lehrerseminar</td>
<td>5278.84</td>
<td>235.08</td>
<td>265.19</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>4</td>
<td>Berufsschule, Maturitätsschule</td>
<td>5893.82</td>
<td>274.52</td>
<td>345.76</td>
</tr>
<tr>
<td>5</td>
<td>22, 23</td>
<td>5</td>
<td>Maturitätsschule, Lehrerseminar</td>
<td>6957.75</td>
<td>291.47</td>
<td>433.38</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>6</td>
<td>Höhere Fach- und Berufsausbildung</td>
<td>6812.49</td>
<td>466.90</td>
<td>678.27</td>
</tr>
<tr>
<td>7</td>
<td>32, 33</td>
<td>7</td>
<td>Höhere Fachschule, Fachhochschule</td>
<td>6812.49</td>
<td>466.90</td>
<td>678.27</td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>8</td>
<td>Universität, Hochschule</td>
<td>6812.49</td>
<td>466.90</td>
<td>678.27</td>
</tr>
</tbody>
</table>

*MATSim code 21 = mean of Microcensus codes 3 and 4

Source: calculated from Swiss Microcensus 2005 (Swiss Federal Statistical Office 2006)

2. **Calibration with average municipality income**: Information on average income by Swiss municipality is available from Beige (2008). For each municipality, we compared agents’ average income according to step 1 with the given municipality’s average income. If differences existed, agents’ incomes were scaled up/down by the difference. In summary, incomes were scaled up by 1,456 CHF (see figure 6.15). This is quite substantial and may be due to the different data sources. It may be also caused by the fact that income by educational qualifications refers to Switzerland overall, while agents in the Greater Zurich scenario are based in municipalities around (wealthy) Zurich.

3. **Application of stochastic spread**: Finally, a random draw of a normal distribution with standard distribution \( s = 0.5 \) was applied to each income of step 2 (s.t. normal distribution truncated at the ordinate, or \( \text{income} \geq 0 \) respectively). The truncation explains the small increase in incomes after step 3.

### 6.4.2 Season ticket ownership

We had planned to determine season ticket ownership through an MNL model from persons’ additional socio-demographic attributes, such as age, gender, income, etc. However, the MNL model showed no explanatory power (19 of 20 parameters not significant). Instead, we devel-
Choosing the Daily Schedule: Expanding Activity-Based Travel Demand Modelling

Figure 6.15: Generating agents’ incomes: summary of average incomes per step

<table>
<thead>
<tr>
<th></th>
<th>By education</th>
<th>By education</th>
<th>Adjusted to municipalities</th>
<th>After normal distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average incomes in CHF</strong></td>
<td>5,043</td>
<td>4,875</td>
<td>1,456</td>
<td>6,352</td>
</tr>
<tr>
<td>Microcensus 2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greater Zurich scenario</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

opened a model of socio-economic group probabilities. Based on Microcensus data, the model defines the probability of holding a season ticket for socio-economic groups of people (e.g., between 20 and 29 years old, female, owning a driver’s license, with a car always available, and with a monthly household income of between 4,000 and 7,999 CHF). Agents are then assigned a season ticket according to these probabilities. The model considers five socio-economic attributes:

- **Age**: discretized into eight classes of 1-9 years, 10-19 years, ..., 70 years and above.
- **Gender**: male or female.
- **Driver’s license ownership**: yes or no.
- **Car availability**: always, sometimes, never.
- **Monthly household income**: discretized into four classes of 0-3,999 CHF, 4,000-7,999 CHF, 8,000-11,999 CHF, 12000 CHF and above.

The attributes and their classes sum up to $8 \cdot 2 \cdot 2 \cdot 3 \cdot 4 = 384$ possible socio-economic groups of people. Microcensus data covers 195 of the 384 groups[26]. 168,371 out of 172,598 agents of the Greater Zurich scenario are covered by the 195 groups. They are provided with a season ticket type according to the groups’ probabilities. The remaining agents are assigned according to Microcensus’ overall probability of holding a season ticket type, disregarding any socio-economics. Figure 6.16 illustrates overall fit of the model.

[26] 181 groups are not covered at all (e.g., a 1-9 years old child with driver’s license, etc.), and eight groups consist of only one single person, not considered explanatory.
Figure 6.16: Comparison of probabilities of season ticket ownership

[Graph showing season ticket ownership in % of population]

6.5 Discussion

MATSim’s new utility function features an asymmetric S-shaped curve with an inflection point, as presented by [Joh (2004)]. The update is necessary, because the existing function’s log form is problematic in combination with PlanomatX and schedule recycling. When number of schedule activities is a dimension of the learning process, the log form leads to numerous very short activities due to its decreasing marginal utility. Beyond specification of the new functional form, we have disaggregated some attributes (e.g., travel time by mode) and introduced new attributes (e.g., travel cost, socio-economic attributes).

Utility function parameters have been empirically estimated using an enhanced Multinomial Logit (MNL) model. The model features a dissimilarity attribute in the systematic part of the function, reflecting structural dissimilarity between schedules following the path-size logit or C-logit approach. However, comparison of simulation results with census and traffic count data shows that estimated utility function does not satisfactorily reproduce reality. Our choice set misses information on order of activities within the chain. Moreover, the MNL model’s dissimilarity term effectively contributes to the estimation model fit, but biases schedule optimization, as it is no longer part of the operational utility function when run with MATSim. We have solved the problems through a manual calibration of the utility function parameters. Running with schedule recycling, the calibrated utility function reproduces most schedule characteristics well (activity frequencies and durations, trip numbers and mode shares). Replication of the specific activity chain distributions is an exception. Here, both PlanomatX and schedule recycling still lack some precision. Nonetheless, observed and simulated traffic counts correspond well.

Finally, the two attributes monthly household income and season ticket ownership were not default attributes of the greater Zurich synthetic population and had to be generated. The income attribute was derived from municipality and education data. The definition of socio-economic groups, drawn from Microcensus, helped determine the season ticket attribute.
Chapter 7

Conclusion and outlook

This thesis presents the new scheduling algorithms PlanomatX and TimeModechoicer, and a new schedule recycling concept. All are part of the agent and activity-based transport simulation MATSim. PlanomatX generates comprehensively optimized all-day schedules, i.e. optimal combinations of a schedule’s activity chain (number, type and sequence of activities), activity timings, and location, mode and route choices. It is based on Tabu Search. Tests were conducted on a small, hypothetical chessboard scenario, as well as on the large-scale greater Zurich scenario, with more than 170,000 agents. The tests show that PlanomatX successfully optimizes agents’ schedules. In comparison with joint application of MATSim’s existing replanning strategies (base test), PlanomatX achieved significantly higher utility scores. This was possible because the activity chain was also a dimension of MATSim’s co-evolutionary learning process.

However, PlanomatX requires disproportional runtimes. The largest runtime contributor turned out to be Planomat, MATSim’s existing module for optimization of activity timings and mode choices. We have therefore designed TimeModeChoicer, which replicates Planomat’s functionality but is based on Tabu Search like PlanomatX. TimeModeChoicer runs about six times faster than Planomat without quality loss. TimeModeChoicer can be run as a MATSim stand-alone replanning strategy, but is particularly designed for integration with PlanomatX.

The new concept of schedule recycling allows further reduction of simulation runtimes. Schedule recycling avoids running PlanomatX for each agent individually. Instead, it re-uses schedules of optimized agents for other non-optimized agents, significantly reducing simulation runtimes with acceptable quality losses. Figure 7.1 summarizes utility scores and runtimes of PlanomatX, schedule recycling and the reference base test.

The new scheduling algorithms required enhancement of MATSim’s existing utility function for activities performance. The existing function, in combination with PlanomatX, would have
led to schedules with numerous very short activities, due to decreasing marginal utility of its log-form. We replaced the function by an asymmetric S-shaped curve with an inflection point, as presented by Joh (2004). The new function can cope with a flexible number of activities in the schedule as it formulates an optimal activity duration with its functional form.

In addition to modifying the utility function’s form, we disaggregated some function attributes (e.g., travel time by mode) and incorporated new attributes (e.g., travel cost, socio-economic attributes). We further empirically estimated parameters of the new utility function using a Multinomial Logit (MNL) model enhanced by a dissimilarity term in the utility function’s systematic part. However, comparison of simulation results with census and traffic count data shows that the estimated utility function does not satisfactorily reproduce reality. Analysis of differences between simulation and reported/observed data suggests that our choice set misses information on order of activities within the chain. The analysis further suggests that MNL model’s dissimilarity term clearly contributes to estimation model fit, but biases schedule optimization, because it is no longer part of the operational utility function when run with MATSim. We solved those problems through a manual calibration of utility function parameters. Calibrated utility function effectively reproduces most schedule characteristics (activity frequencies and durations, trip numbers and mode shares), with the exception of specific activity chain distributions. Here, both PlanomatX and schedule recycling still lack some precision.

Test and calibration results point to several areas with potential for development:
Further alignment of PlanomatX and schedule recycling: PlanomatX and schedule recycling are heuristic, non-exact algorithms. They produce slightly different optimization results, particularly in their activity chain distributions. However, it would be ideal if both resulted in identical optimization results, implying that schedule recycling needs to further improve. A core lever may be refinement of schedule recycling’s distance metric definition. This could require considering more discrete agent attributes (step D/F3, see figure 5.2) or improving the distance metric algorithm (step C, see figure 5.1): The current algorithm is straightforward, but (too) simple. A better algorithm would avoid limitation of discrete offset moves and allow for a continuous metric definition. Another more specific lever could be alignment of schedule recycling’s travel distance behaviour. Schedule recycling does not reduce travel distance as well as PlanomatX (irrespective of the fact that both underestimate reported travel distance; see next bullet point). Travel distance is closely linked with agents’ location choices and, thus, schedule recycling’s customization operator (step D/F7, see figure 5.2) may be a point of interest. For instance, it might be possible to run several iterations of location choice and select that combination of locations offering maximum utility - which is equal to minimum distance, s.t. the underlying utility function.

Integration of attraction-driven location choice: [Horni et al. (2009)] present an approach based not only on Hägerstrand’s time-space prisms, but also evaluating attraction of locations (common distance to leisure locations, shopping store size and load density) within these prisms. The incorporation of such a location choice might correct both PlanomatX’s and schedule recycling’s under-estimation of overall distance travelled per schedule.

Improved modelling of public transport: Flat estimation of public transport travel time - as 1.8 times the car free flow travel time - is likely to bias optimization results. Indicators of bias include under-use of public transport at low distances (up to about 2 km) and at long distances (beyond 50 km). Travel time matrices or a frequency-/timetable-based assignment model could improve public transport modelling.

Integration of household interactions: Many activity-based travel demand models described in section 2.2.4 consider interaction among household members, e.g., allocation of maintenance work. The relevance of household interaction to activity scheduling is backed up by numerous publications (see, for instance, special issue of Transportation journal, Bhat and Pendyala [2005]). MATSim still handles only individuals and does not capture those household interactions. Therefore, integration of household interaction will be very important to MATSim, and give PlanomatX and schedule recycling the opportunity to tune their activity scheduling.

Refined estimation of utility function: Last, but not least, the empirical estimation of

---

1 A timetable-based public transport assignment module is being developed for MATSim.
MATSim’s new utility function will require more effort. Two obvious courses of action involve revising choice set structure and clearing dissimilarity bias. The former may imply capturing schedules’ activity orders in choice set alternatives. The latter may require either investing in a more complex discrete choice model or considering the dissimilarity attribute in MATSim’s operational utility function. In the interim, one possibility would involve applying a more formal approach to calibration of the utility function, for instance, neural networks. Regarding utility function specification, the linear form for travel performance may be reconsidered. Positive car travel cost and public transport travel time parameters indicate that travel may have some innate beneficial value that could also be dependent on actual distance travelled. Finally, it might be worthwhile to include some additional monetary attributes in MATSim’s utility function: for instance, parking fees, tolls and activity expenditure/income. However, these attributes will require more data collection.

PlanomatX and schedule recycling show great potential as future tools of travel demand appraisal. Given further adjustment of underlying utility function parameters, PlanomatX and schedule recycling will be capable of replicating travellers’ behaviour, allowing for detailed investigation of reactions to, for instance, changes in infrastructure and policy. Employed jointly with MATSim’s other simulation modules, they will help researchers and practitioners to take full advantage of activity-based approach to travel demand modelling.

\(^2\)For instance, a nested logit (NL) or mixed multinomial logit (MMNL) model.
Bibliography


Srinivasan, S. and S. R. Athuru (2005) Analysis of within-household effects and between-

Srinivasan, S. and C. R. Bhat (2005) Modeling household interactions in daily in-home and


