Dynamic Programming Approaches for Estimating and Applying Large-scale Discrete Choice Models

par

Anh Tien Mai

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Décembre 2015

©Anh Tien Mai, 2015
“Logic will get you from A to B. Imagination will take you everywhere.”

Albert Einstein
Abstract

People go through their life making all kinds of decisions, and some of these decisions affect their demand for transportation, for example, their choices of where to live and where to work, how and when to travel and which route to take. Transport related choices are typically time dependent and characterized by large number of alternatives that can be spatially correlated. This thesis deals with models that can be used to analyze and predict discrete choices in large-scale networks. The proposed models and methods are highly relevant for, but not limited to, transport applications.

We model decisions as sequences of choices within the dynamic discrete choice framework, also known as parametric Markov decision processes. Such models are known to be difficult to estimate and to apply to make predictions because dynamic programming problems need to be solved in order to compute choice probabilities. In this thesis we show that it is possible to explore the network structure and the flexibility of dynamic programming so that the dynamic discrete choice modeling approach is not only useful to model time dependent choices, but also makes it easier to model large-scale static choices.

The thesis consists of seven articles containing a number of models and methods for estimating, applying and testing large-scale discrete choice models. In the following we group the contributions under three themes: route choice modeling, large-scale multivariate extreme value (MEV) model estimation and nonlinear optimization algorithms.

Five articles are related to route choice modeling. We propose different dynamic discrete choice models that allow paths to be correlated based on the MEV and mixed logit models. The resulting route choice models become expensive to estimate and we deal with this challenge by proposing innovative methods that allow to reduce the estimation cost. For example, we propose a decomposition method that not only opens up for possibility of mixing, but also speeds up the estimation for simple logit models, which has implications also for traffic simulation. Moreover, we compare the utility maximization and regret minimization decision rules, and we propose a misspecification test for logit-based route choice models.

The second theme is related to the estimation of static discrete choice models with large choice sets. We establish that a class of MEV models can be reformulated as dynamic discrete choice models on the networks of correlation structures. These dynamic models can then be estimated quickly using dynamic programming techniques and an efficient nonlinear optimization algorithm.

Finally, the third theme focuses on structured quasi-Newton techniques for estimating discrete choice models by maximum likelihood. We examine and adapt switching methods that can be
easily integrated into usual optimization algorithms (line search and trust region) to accelerate the estimation process.

The proposed dynamic discrete choice models and estimation methods can be used in various discrete choice applications. In the area of big data analytics, models that can deal with large choice sets and sequential choices are important. Our research can therefore be of interest in various demand analysis applications (predictive analytics) or can be integrated with optimization models (prescriptive analytics). Furthermore, our studies indicate the potential of dynamic programming techniques in this context, even for static models, which opens up a variety of future research directions.

**Keywords:** discrete choice modeling, route choice modeling, dynamic programming, dynamic discrete choice models, misspecification tests, nonlinear optimization algorithms, maximum likelihood estimation, prediction.
Résumé

Les gens consacrent une importante part de leur existence à prendre diverses décisions, pouvant affecter leur demande en transport, par exemple les choix de lieux d’habitation et de travail, les modes de transport, les heures de départ, le nombre et type de voitures dans le ménage, les itinéraires, ... Les choix liés au transport sont généralement fonction du temps et caractérisés par un grand nombre de solutions alternatives qui peuvent être spatialement corrélées. Cette thèse traite de modèles pouvant être utilisés pour analyser et prédire les choix discrets dans les applications liées aux réseaux de grandes tailles. Les modèles et méthodes proposées sont particulièrement pertinents pour les applications en transport, sans toutefois s’y limiter.

Nous modélisons les décisions comme des séquences de choix, dans le cadre des choix discrets dynamiques, aussi connus comme processus de décision de Markov paramétriques. Ces modèles sont réputés difficiles à estimer et à appliquer en prédiction, puisque le calcul des probabilités de choix requiert la résolution de problèmes de programmation dynamique. Nous montrons dans cette thèse qu’il est possible d’exploiter la structure du réseau et la flexibilité de la programmation dynamique afin de rendre l’approche de modélisation dynamique en choix discrets non seulement utile pour représenter les choix dépendant du temps, mais également pour modéliser plus facilement des choix statiques au sein d’ensembles de choix de très grande taille.

La thèse se compose de sept articles, présentant divers modèles et méthodes d’estimation, leur application ainsi que des expériences numériques sur des modèles de choix discrets de grande taille. Nous regroupons les contributions en trois principales thématiques: modélisation du choix de route, estimation de modèles en valeur extrême multivariée (MEV) de grande taille et algorithmes d’optimisation non-linéaire.

Cinq articles sont associés à la modélisation de choix de route. Nous proposons différents modèles de choix discrets dynamiques permettant aux utilités des chemins d’être corrélées, sur base de formulations MEV et logit mixte. Les modèles résultats devenant coûteux à estimer, nous présentons de nouvelles approches permettant de diminuer les efforts de calcul. Nous proposons par exemple une méthode de décomposition qui non seulement ouvre la possibilité d’estimer efficacement des modèles logit mixte, mais également d’accélérer l’estimation de modèles simples comme les modèles logit multinomiaux, ce qui a également des implications en simulation de trafic. De plus, nous comparons les règles de décision basées sur le principe de maximisation d’utilité de celles sur la minimisation du regret pour ce type de modèles. Nous proposons finalement un test statistique sur les erreurs de spécification pour les modèles de choix de route basés sur le logit multinomial.

Le second thème porte sur l’estimation de modèles de choix discrets statiques avec de grands ensembles de choix. Nous établissons que certains types de modèles MEV peuvent être reformulés comme des modèles de choix discrets dynamiques, construits sur des réseaux de structure
de corrélation. Ces modèles peuvent alors être estimées rapidement en utilisant des techniques de programmation dynamique en combinaison avec un algorithme efficace d’optimisation non-linéaire.

La troisième et dernière thématique concerne les algorithmes d’optimisation non-linéaires dans le cadre de l’estimation de modèles complexes de choix discrets par maximum de vraisemblance. Nous examinons et adaptons des méthodes quasi-Newton structurées qui peuvent être facilement intégrées dans des algorithmes d’optimisation usuels (recherche linéaire et région de confiance) afin d’accélérer le processus d’estimation.

Les modèles de choix discrets dynamiques et les méthodes d’optimisation proposés peuvent être employés dans diverses applications de choix discrets. Dans le domaine des sciences de données, des modèles qui peuvent traiter de grands ensembles de choix et des ensembles de choix séquentiels sont importants. Nos recherches peuvent dès lors être d’intérêt dans diverses applications d’analyse de la demande (analyse prédictive) ou peuvent être intégrées à des modèles d’optimisation (analyse prescriptive). De plus, nos études mettent en évidence le potentiel des techniques de programmation dynamique dans ce contexte, y compris pour des modèles statiques, ouvrant la voie à de multiples directions de recherche future.

**Mots-clés:** modélisation, choix discrets, choix d’itinéraire, programmation dynamique, choix discrets dynamiques, tests de spécification, estimation du maximum de vraisemblance, algorithmes d’optimisation, prédiction.
Acknowledgements

First of all, I do not know how to express my gratitudes to my thesis supervisor, Prof. Emma Frejinger. Her motivation, original ideas, enthusiasm, curiosity and rigor have inspired me throughout the 3 years of my Ph.D. studies. To me, she is not only a thesis supervisor but also a dear sister who helped me to improve myself from little things. During these years, she taught me how to organize my ideas, to write a complete scientific document and how to explain my thoughts clearly and convincingly to other people. Moreover, she has introduced me to a network of brilliant and interesting people over the world. Everything she has done for me has not only directly contributed to the work presented in this thesis, but has also been a great gift for me personally. I am forever grateful to her.

Prof. Fabian Bastin is my co-supervisor who has always been encouraging and supporting me throughout my 5 years in Montréal. He has always listened to me carefully and analyzed my ideas by the best of his knowledge and belief. His ideas and suggestions have meant a lot, not only to my research but also to my personal life. I am very thankful to him.

I am very grateful to Prof. Mogens Fosgerau who has made very important contributions to this thesis. He is the co-author of two articles and has directly or indirectly contributed to the other papers presented in this thesis. I have also benefited a lot from his comments, suggestions as well as point of view. Moreover, most of the work presented in this thesis is motivated by the route choice modeling idea from his paper published in 2013 in Transportation Research Part B. I have also been collaborating with Prof. Song Gao, Jing Ding and Xinlian Yu during these years, and I thank them for having spent the time to work with me on the routing policy choice problem.

I am grateful to Prof. Moshe Ben-Akiva for giving me the opportunity to visit his Intelligent Transportation Systems laboratory at MIT. I have been working intensively on the writing of this thesis during my stay at MIT, and have benefited from the stimulating atmosphere that I experienced there.

I am thankful to the jury members of my thesis: Professors Pierre L’Ecuyer, Patrice Marcotte, Hani Mahmassani, for the valuable discussions and their constructive feedbacks.

This research was supported by the National Sciences and Engineering Research Council of Canada, discovery grant 435678-2013. I am also very thankful to Département de l’Informatique et de Recherche Opérationnelle (DIRO), Centre Inter-universitaire de Recherche sur les Réseaux d’Entreprise, la Logistique et le Transport (CIRRELT), Faculté des Études Supérieures et Postdoctorales (FESP - Université de Montréal) for their financial supports during my studies.

I am more than grateful to my parents, parents-in-law as well as my two sisters for their continuous support and love. Having them, I always feel safe and warm.
Finally, the last thank goes to Thuy Anh Ta, my sweet wife and also my labmate, for having been beside me and constantly encouraged me all the time. No words can enough express her important contributions to this work as well as to my life.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Résumé</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiv</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>xvi</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Background, Motivation and Objectives</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Contributions and Outline</td>
<td>3</td>
</tr>
<tr>
<td><strong>2 Literature Review</strong></td>
<td>6</td>
</tr>
<tr>
<td>2.1 Discrete Choice Analysis</td>
<td></td>
</tr>
<tr>
<td>2.1.1 Decision Rules</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Multinomial Logit</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3 Multivariate Extreme Value Models</td>
<td>8</td>
</tr>
<tr>
<td>2.1.4 Mixed Multinomial Logit</td>
<td>9</td>
</tr>
<tr>
<td>2.1.5 Sampling of Alternatives</td>
<td>10</td>
</tr>
<tr>
<td>2.1.6 Dynamic Discrete Choice Models</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Maximum Likelihood Estimation</td>
<td>14</td>
</tr>
<tr>
<td>2.2.1 Log-Likelihood Function and Properties</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2 Maximizing the Log-Likelihood Function</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3 Estimating Dynamic Discrete Choice Models</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Statistical Tests of Model Misspecification</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Route Choice Analysis using Discrete Choice Models</td>
<td>22</td>
</tr>
<tr>
<td>2.4.1 Path-based Models with Sampled Choice Sets</td>
<td>24</td>
</tr>
<tr>
<td>2.4.2 Link-based Recursive Logit Model</td>
<td>25</td>
</tr>
<tr>
<td>2.4.3 Illustrative Example</td>
<td>28</td>
</tr>
<tr>
<td><strong>3 Nested Recursive Logit Model</strong></td>
<td>31</td>
</tr>
</tbody>
</table>
3.1 Introduction .................................................. 32
3.2 The Nested Recursive Logit Model .......................... 34
3.3 Illustrative Examples and Substitution Patterns .......... 37
3.4 Computation of the Value Functions ......................... 41
3.5 Maximum Likelihood Estimation ............................ 42
3.6 Numerical Results ........................................... 44
3.6.1 Model Specifications ..................................... 45
3.6.2 Estimation Results ........................................ 46
3.6.3 Prediction Results ........................................ 48
3.7 Conclusion .................................................... 49

4 Integrating Correlation Structures for a Generalized Recursive Model 51
4.1 Introduction .................................................. 52
4.2 Generalized Recursive Route Choice Model ................. 55
4.3 Method of Integrating Correlation Structures .............. 58
4.3.1 Integrated Network ....................................... 59
4.3.2 NRL Model on the Integrated Network ................. 61
4.3.3 Similarities between the NRL on Network $G$* and RNMEV on Network $G$ 62
4.4 Maximum Likelihood Estimation ............................ 62
4.4.1 Computation of the Value Functions ..................... 63
4.4.2 Maximum Likelihood Estimation ......................... 64
4.5 Recursive Cross-nested Logit Model ......................... 65
4.6 Application .................................................... 67
4.6.1 Model Specification ...................................... 67
4.6.2 Estimation Results ........................................ 69
4.6.3 Prediction Results ........................................ 71
4.7 Conclusion .................................................... 72

5 A Decomposition Method for Estimating Recursive Logit Based Models 74
5.1 Introduction .................................................. 75
5.2 Recursive Logit ............................................... 77
5.3 Decomposition Method ...................................... 78
5.4 Mixed Recursive Logit ....................................... 81
5.5 Modeling Correlations with Subnetwork Components .... 83
5.6 Numerical Results ........................................... 84
5.6.1 Specification of Models .................................. 84
5.6.2 Estimation Results ........................................ 86
5.6.3 Prediction Results ........................................ 88
5.6.4 Evaluation of Computational Time for the Decomposition Method 88
5.7 Conclusion .................................................... 90

6 Comparing Regret Minimization and Utility Maximization 91
6.1 Introduction .................................................. 92
6.2 Random utility maximization and random regret minimization models 94
6.3 Random regret recursive logit models ....................... 96
6.4 Link regret formulations .................................... 99
6.5 Competitive RUM models .................................. 101
6.6 Illustrative example ...................................................... 101
6.7 Maximum likelihood estimation .................................... 103
6.8 Numerical results ......................................................... 105
  6.8.1 Model specifications ................................................. 105
  6.8.2 Estimation results .................................................... 108
  6.8.3 Prediction results .................................................... 111
6.9 Conclusion ................................................................. 113

7 A Misspecification Test for Logit Based Route Choice Models 115
  7.1 Introduction ............................................................. 116
  7.2 Maximum Likelihood Estimation and Information Matrix Test 118
  7.3 Route Choice Models .................................................. 119
    7.3.1 Path-based MNL Models ......................................... 120
    7.3.2 Recursive Logit .................................................. 121
  7.4 Monte Carlo Experiment .............................................. 123
    7.4.1 Experimental Setup ............................................. 124
    7.4.2 Numerical Results .............................................. 126
  7.5 Numerical Results using Real Data ................................ 128
  7.6 Remarks on Computational Time .................................... 129
  7.7 Conclusions ............................................................ 131

8 Estimating Large-scale MEV Models with Dynamic Programming 133
  8.1 Introduction ............................................................. 134
  8.2 The Network MEV Model ............................................. 136
  8.3 Formulating the Network MEV Model using the Dynamic Discrete Choice Frame-
      work .................................................................. 137
  8.4 Illustrative Example .................................................... 140
  8.5 Maximum Likelihood Estimation .................................... 141
    8.5.1 Solving the Value Functions ................................... 142
    8.5.2 Choice Probabilities ............................................. 143
    8.5.3 Log-likelihood Function and Non-linear Optimization 143
  8.6 Application to Large Problems ..................................... 144
    8.6.1 Experimental Setup ............................................. 145
    8.6.2 Computational Time Results .................................. 146
  8.7 Comparisons of Computational Time .............................. 148
  8.8 Conclusion ............................................................... 150

9 Structured Quasi-Newton Techniques for Maximum Likelihood Estimation 152
  9.1 Introduction ............................................................. 153
  9.2 Structured Quasi-Newton Methods ................................ 154
  9.3 Maximum Likelihood Estimation ................................... 156
  9.4 Optimization Algorithms ............................................ 157
    9.4.1 Line Search Methods .......................................... 158
    9.4.2 Trust Region Methods ........................................... 159
  9.5 Hessian Approximations .............................................. 159
    9.5.1 Secant Approximations ......................................... 160
    9.5.2 BHHH Approximation .......................................... 161
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Small network</td>
<td>28</td>
</tr>
<tr>
<td>2.2</td>
<td>Path utilities</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Illustration of notation</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Classic three paths example network</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Example network with perfect nested structure</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>Example network with cross-nested structure</td>
<td>39</td>
</tr>
<tr>
<td>3.5</td>
<td>Cross-nested structure from the Link-nested logit model</td>
<td>40</td>
</tr>
<tr>
<td>3.6</td>
<td>Histogram of $\mu_k$ and $\ln \phi_{ka}$ for NRL-LS</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Average of the test error values over holdout samples</td>
<td>50</td>
</tr>
<tr>
<td>4.1</td>
<td>Illustration of notation</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>A network of correlation structure at state $k$</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Original and integrated network</td>
<td>59</td>
</tr>
<tr>
<td>4.4</td>
<td>A cross-nested structure at link $k$</td>
<td>66</td>
</tr>
<tr>
<td>4.5</td>
<td>Average of the test error values over holdout samples</td>
<td>72</td>
</tr>
<tr>
<td>5.1</td>
<td>A small network with multiple destinations</td>
<td>77</td>
</tr>
<tr>
<td>5.2</td>
<td>Borlänge subnetworks (Frejinger and Bierlaire, 2007)</td>
<td>85</td>
</tr>
<tr>
<td>5.3</td>
<td>Final log-likelihood over 10 MC repetitions</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>Average of the test error values over holdout samples</td>
<td>89</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of notation</td>
<td>97</td>
</tr>
<tr>
<td>6.2</td>
<td>Classic three paths example network</td>
<td>102</td>
</tr>
<tr>
<td>6.3</td>
<td>Path probabilities</td>
<td>103</td>
</tr>
<tr>
<td>6.4</td>
<td>$P_1$ as function of $\beta$ and $\delta$</td>
<td>104</td>
</tr>
<tr>
<td>6.5</td>
<td>Average of the test error values over holdout samples</td>
<td>114</td>
</tr>
<tr>
<td>7.1</td>
<td>Box plot of number of paths in $C_n$ for $n = 1, \ldots, 1832$ and over 100 samples</td>
<td>125</td>
</tr>
<tr>
<td>8.1</td>
<td>A cross-nested structure</td>
<td>141</td>
</tr>
<tr>
<td>8.2</td>
<td>A cross-nested structure</td>
<td>146</td>
</tr>
<tr>
<td>8.3</td>
<td>A three-levels cross-nested structure</td>
<td>146</td>
</tr>
<tr>
<td>C.1</td>
<td>Cumulative distribution of $p$-values used to assess the number of type I errors, RL-LS model</td>
<td>196</td>
</tr>
<tr>
<td>C.2</td>
<td>Cumulative distribution of $p$-values used to assess the number of type I errors, RL and PL models</td>
<td>197</td>
</tr>
<tr>
<td>C.3</td>
<td>Cumulative distribution of $p$-values used to assess the number of type II errors, RL and PL models</td>
<td>198</td>
</tr>
</tbody>
</table>
D.1 Tree $T$ .............................................................. 204
# List of Tables

2.1 Numerical results when the true choice sets are \((U_1, U_2)\) ........................................ 29  
2.2 Numerical results when the true choice sets are \((C^3_1, C^3_2)\) .................................. 29  
3.1 Change in probability when link is removed (example network with perfect nested structure) ................................................................. 38  
3.2 Change in probability when link is removed (example network with perfect nested structure with link from B to C) ........................................ 39  
3.3 Change in probability when link is removed (example network with cross-nested structure) ................................................................. 40  
3.4 Path probabilities comparison ................................................................................. 40  
3.5 Estimation results .................................................................................................... 48  
3.6 Likelihood ratio test results ..................................................................................... 48  
3.7 Average of test error values over 40 holdout samples .............................................. 49  
4.1 Estimation results .................................................................................................... 70  
4.2 Likelihood ratio test results ..................................................................................... 71  
4.3 Average of test error values over 20 holdout samples .............................................. 72  
5.1 Estimation results with IRN and CRN ....................................................................... 83  
5.2 Averaged final log-likelihood values ...................................................................... 87  
5.3 Likelihood ratio test results ..................................................................................... 87  
5.4 Average of test error values over 40 holdout samples .............................................. 88  
5.5 Comparison of computational times (in seconds) .................................................... 89  
6.1 Estimation results .................................................................................................... 109  
6.2 Final log-likelihood values of the RUM-based RL and NRL models ..................... 110  
6.3 Final log-likelihood values of the RL models with the RRM and CRUM frameworks 110  
6.4 Average of out-of-sample error values for the RL and NRL models .................... 112  
6.5 Average of out-of-sample error values for the RRM and CRUM models ............. 113  
7.1 Number of type I errors, 100 repetitions, 0.05 significance level ......................... 127  
7.2 Number of type II errors (100 repetitions), 0.05 significance level, the results based on size-corrected critical values are given in parenthesis ................... 128  
7.3 Estimation results for real data ............................................................................... 129  
7.4 Test statistic values for IM, IIA and McTr tests for models estimated on real data 129  
7.5 Computational time, in hours, for estimation and testing ....................................... 131  
8.1 Models and simulated data sets ............................................................................... 145  
8.2 Computational time (in seconds) ............................................................................. 148  
8.3 Computational time comparisons (in seconds) ....................................................... 150  

xiv
8.4 Computational time (in seconds) comparisons of matrix and recursive operations 151

9.1 Data sets ................................................................. 171
9.2 Summary of stopping conditions ........................................ 173
9.3 Performance comparison with mixed logit models ................. 174
9.4 Performance comparison with MNL route choice models ......... 174
9.5 Rate of successful simulations for a difficult case (ILN) .......... 176
9.6 Average number of switches per estimation in the switching methods ...... 176

B.1 Estimation results for the mixed RL with random parameters .............. 190
B.2 Estimation results for the mixed RL model with EC ..................... 191
B.3 Estimation results for the mixed RL model with EC and RP ............. 192
Abbreviations

ACRUM  Averaged Competitive Random Utility Maximization
ARRM  Averaged Random Regret Minimization
BHHH  Berndt-Hall-Hall-Hausman approximation
BFGS  Broyden-Fletcher-Goldfarb-Shanno (or rank-2) approximation
CF  Commonality Factor
CPGF  Choice Probability Generating Function
CRN  Common Random Numbers
CRUM  Competitive Random Utility Maximization
DeC  Decomposition method
EC  Error Component
ECRUM  Extended Competitive Random Utility Maximization
ERRM  Extended Random Regret Minimization
EPS  Extended Path Size
GEV  Generalized Extreme Value
GPS  Global Positioning System
GRRM  Generalized Random Regret Minimization
IIA  Independence from Irrelevant Alternatives
i.i.d.  independent and identically distributed
IRN  Independent Random Numbers
IM  Information Matrix
LNL  Link-nested Logit
LL  Log-likelihood
LS  Link Size
McTr  McFadden and Train’s Lagrange multiplier test
MEV  Multivariate Extreme Value
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMNL</td>
<td>Mixed MultiNomial Logit</td>
</tr>
<tr>
<td>MNL</td>
<td>MultiNomial Logit</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>MRL</td>
<td>Mixed Recursive Logit</td>
</tr>
<tr>
<td>NFXP</td>
<td>Nested Fixed Point algorithm</td>
</tr>
<tr>
<td>NRL</td>
<td>Nested Recursive Logit</td>
</tr>
<tr>
<td>OD</td>
<td>Origin-Destination</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PS</td>
<td>Path Size</td>
</tr>
<tr>
<td>PSL</td>
<td>Path Size Logit</td>
</tr>
<tr>
<td>RCNL</td>
<td>Recursive Cross-nested Logit model</td>
</tr>
<tr>
<td>RL</td>
<td>Recursive Logit</td>
</tr>
<tr>
<td>RNMEV</td>
<td>Recursive Network MEV model</td>
</tr>
<tr>
<td>RP</td>
<td>Reveled Preference</td>
</tr>
<tr>
<td>RRM</td>
<td>Random Regret Minimization</td>
</tr>
<tr>
<td>RUM</td>
<td>Random Utility Maximization</td>
</tr>
<tr>
<td>SR1</td>
<td>Symmetric Rank-1 approximation</td>
</tr>
</tbody>
</table>
To my wife, our parents and sisters
Chapter 1

Introduction

1.1 Background, Motivation and Objectives

Transportation of persons and goods is indispensable for economic growth but also has negative impacts such as noise, congestion and pollution. In order to ensure a sustainable development, transport systems need to be designed such that the negative impacts are minimized while supporting growth in the economy. Transport planning models are used for this purpose and an essential part of these models is an accurate characterization of the demand. This thesis focuses on demand models that can be used to predict travelers’ behavior in transport systems.

Persons’ transport demand is derived from a number of choices. For instance, where they choose to live and work, where to shop, which transport modes to use and which route to take. Choice problems that are transport related are often characterized by a large number of alternatives that have geographical locations, which induce spatial correlation. The models and methods proposed in this thesis are developed for this type of problems and focus on large-scale network-based choices using the discrete choice framework. We note however that the models are general and not limited to transport applications.

We focus on models for the choice of path in a transport network (also known as the route choice problem), and complex static discrete choice models with large number of alternatives. The route choice problem deals with identifying which route a given traveler would take to go from a location to another in a transport network. Route choice models are important in the sense that they allow to access travelers’ preferences of various route characteristics (e.g. travel time, travel cost, route length, number of crossings, number of turns) as well as the affect of travelers’ characteristics to the choice of route (e.g. age, income, gender). The parameter estimates from route choice models can be used to analyze preferences towards certain infrastructures or evaluate the value of travel time. Moreover, route choice models may be useful in other
transportation applications such as traffic simulation and activity choice modeling (paths are defined based on activity networks).

In transportation applications, due to the large number of elements in a real transport network (e.g. numbers of links, nodes and paths), many studies take an approach that consists of sampling alternatives from the network and estimate models based on these samples. This approach can be difficult in practice and it is not clear how to sample alternatives (and correct probabilities accordingly) for prediction. This thesis originates from the ideas introduced by Fosgerau et al. (2013a). They propose the recursive logit model that is based on the dynamic discrete choice modeling framework (also known as parametric Markov decision processes) that allows to model the choice of path in a network without sampling any choice sets. Dynamic discrete choice models are typically used to capture forward-looking behavior of decision makers, i.e. at each choice stage they make a choice with an eye towards how the decision will impact future outcomes. These models are more costly to estimate and to apply than static ones because they require solving dynamic programming problems in order to compute choice probabilities. Fosgerau et al. (2013a) show that the dynamic programming problem is easy to solve if path utilities are not correlated. In this thesis we explore network structures to model large-scale choice problems where utilities can be correlated. We define four main objectives in the following (the text in bold indicates the keywords).

The first objective is to design models and methods for route choice analysis using revealed preferences data. More precisely, we aim at taking the advantage of the dynamic discrete choice framework to design network-based route choice models and corresponding estimation methods that allows to conveniently capture the correlation between path alternatives in real transport networks, and make the proposed models useful for both estimation and prediction. Moreover, we are also interested in statistical tests that can be used to assess the models.

The second objective is to develop methods for quickly estimating static discrete choice model with complex correlation structures. This objective is motivated by the increasing availability of revealed preferences data of large-scale problems through, e.g., passive monitoring and registers.

Maximum likelihood estimation is widely used in discrete choice for identifying parameter values of models. The third objective is to develop nonlinear optimization algorithms that allow to improve the convergence rate.

Finally, besides methodological contributions, this thesis has the objective to be application oriented. The last goal is therefore to use real networks and real or large simulated datasets to illustrate and validate proposed models and methods. Finally, we aim at providing optimization packages for the estimation of large-scale discrete choice models for other researchers and practitioners.
The work under these objectives has resulted in a series of articles as we outline in the following section.

1.2 Thesis Contributions and Outline

This thesis makes a number of important contributions to demand modeling in networks. We group them under three themes: route choice modeling, large-scale multivariate extreme value (MEV) model estimation, and nonlinear optimization algorithms. Five articles are related to route choice modeling. We propose different dynamic discrete choice models for the route choice problem that allow paths to be correlated based on the nested logit, MEV, mixed logit models, as well as through the random regret minimization decision rule. We design methods that allow to solve the resulting dynamic programming problems in short computational time. The proposed dynamic route choice models can be consistently estimated and used for prediction without sampling of alternatives. We also propose a misspecification test for logit based route choice models, and compare different decision rules using this type of models. It is important to note that, in this thesis, we use uni-modal networks and revealed preferences data where the link attributes are assumed to be static and deterministic. The models however can be applied straightforwardly to other types of deterministic networks (e.g. dynamic networks where a state is defined by time and location, multi-modal networks where a state is defined by mode and location). When the link attributes are assumed to be stochastic, the corresponding dynamic route choice models become more complicated to deal with, as the Markov transition probabilities are no longer degenerate. An ongoing work related to this type of network will be discussed in more detail in Section 10 (Conclusion).

One article is related to the second theme. We show that a dynamic programming approach can be used to estimate large-scale MEV models with network-based correlation structures (Daly and Bierlaire, 2006). We show that MEV models can be formulated as dynamic discrete choice models on the networks of correlation structures, and can be quickly estimated using the concept of network flows and the nested fixed point algorithm (Rust, 1987). We validate our approach using simulated data sets and different large networks of correlation structures. We show that the new approach is superior to the classic approach which uses recursive functions. The method therefore can be conveniently applied to any discrete choice model, e.g. combined mode and destination choice.

Under the third theme, we have one article making contributions to optimization algorithms for maximum likelihood estimation. The work is motivated by the fact that, when estimating discrete choice models, it is important to converge in as few iterations as possible when the objective functions are costly, as in the cases considered in this thesis. We examine structured quasi-Newton techniques for maximum likelihood problems and focus on the impact of Hessian
approximation methods on the performance of non-linear optimization algorithms (line search and trust region). We adapt and develop methods that allow to switch between different Hessian approximation matrices. These methods can be easily integrated into the line search and trust region algorithms to improve their performance. We apply our methods for estimating different discrete choice models with real data sets. We also note that the switching approaches may be useful for least-squares problems since the least-squares and maximum likelihood estimation share several similarities in structure.

Finally, the models, optimization methods and tests presented in this thesis are implemented in MATLAB and we share the code freely as open source projects.

The thesis is based on articles where each chapter corresponds to one article. Following the guideline of Université de Montréal, a short description of the paper and the contributions precede each article. In the following we present the outline of the thesis.

**Chapter 2** reviews the literature. We focus on the state-of-the-art of discrete choice and route choice modeling, maximum likelihood estimation, and misspecification tests for discrete choice models.

**Chapter 3** (Mai T., Fosgerau M. and Frejinger E.) present a dynamic discrete choice framework for the route choice problem that allows paths to be correlated by the structure of the transport network. The paper is published in *Transportation Research Part B*, Volume 75(1), p.100-112, 2015.

**Chapter 4** (Mai T.) presents an innovative method for estimating a generalized dynamic route choice model with a network-based correlation structure at each choice stage. The paper has been submitted to *Transportation Research Part B*.

**Chapter 5** (Mai T., Bastin F. and Frejinger E.) presents a decomposition method for estimating a group of complex dynamic route choice models (e.g. mixed route choice models). The paper has been submitted to *EURO Journal on Transportation and Logistics*.

**Chapter 6** (Mai T., Bastin F. and Frejinger E.) compares estimation and prediction results of dynamic route choice models based on different decision rules (random utility maximization and random regret minimization). The paper is currently under review in *Journal of Choice Modeling*.

Chapter 8 (Mai T., Frejinger E., Fosgerau M. and Bastin F.) presents a dynamic programming approach for estimating large-scale static discrete choice models with network-based correlation structures. The paper has been submitted to Transportation Research Part B.

Chapter 9 (Mai T., Bastin F., Toulouse M. and Tao M.) examines nonlinear optimization techniques for maximum likelihood estimation focusing on switching strategies for different Hessian approximations. An early version of the paper was published as a technical report (Mai et al., 2014).

Chapter 10 presents conclusions and future research perspectives that have arisen from the results of this dissertation.

Finally, we include the following four appendices

- Appendix A: The proofs of the two theorems proposed in Chapter 4.
- Appendix B: Supplements to Chapter 5 including detailed estimation results.
- Appendix C: A proof for the validity of the information matrix equality for models with sampled choice sets, detailed second derivatives of the value functions and the numerical results given by a Monte Carlo experiment.
- Appendix D: Supplements to Chapter 8, including detailed proofs of two theorems, first derivatives of the value functions, and the formulations of the elasticities.
Chapter 2

Literature Review

In this chapter, we introduce models and methods which are related to discrete choice analysis with a focus on route choice applications. We assume that the reader is familiar with discrete choice modeling, and refer to Ben-Akiva and Lerman (1985) and Train (2000) for two excellent textbooks on the topic. The reader can also consult Frejinger (2008) for an overview of models and methods for route choice analysis.

In the following section we present a general overview of discrete choice and route choice analysis. In Section 2.1, we present a literature review of discrete choice modeling including the definitions of decision rules, models, sampling of alternatives approaches and dynamic discrete choice models. In Section 2.2, we discuss the maximum likelihood estimation, and Section 2.3 presents different statistical tests for model misspecification. Finally, Section 2.4 focuses on the application of discrete choice models for the route choice problem.

2.1 Discrete Choice Analysis

A discrete choice model describes the choices of decision makers among alternatives under certain general assumptions. A decision maker can be an individual or a group of people (e.g. households, organizations). Alternatives can for instance represent trips, transport modes or locations. A decision maker chooses alternatives from a choice set, and each alternative in the choice set are characterized by a set of attributes that can be alternative and/or individual specific.

A decision is determined by decision rules, which reflect how the decision maker evaluates the alternatives. Most of the discrete choice models in the literature are based on random utility theory, which assumes that the decision makers’ preferences for an alternative is captured by an utility, and the alternative with the highest utility is chosen. Within this framework, the utility
an individual associates with an alternative is typically assumed to be a sum of a deterministic part, which is observed by the analyst, and a random term which is unknown to the analyst. The deterministic part can include attributes of the alternative as well as characteristics of the decision maker while different distributional assumptions can be made on the random term. For example, the random terms can be independent and identically distributed extreme value type I, which leads to the well-known multinomial logit model (McFadden, 1978). The probability that an alternative is chosen by the individual is defined based on the maximum utility theory. These choice probabilities can be used to define a likelihood function over a set of observations, and the model parameters to capture the preferences of the decision maker in the utilities then can be identified via the maximum likelihood estimation.

In the following, we present the state-of-the-art of discrete choice analysis including decision rules and different discrete choice models which are relevant to the models proposed in this thesis. We then present sampling of alternatives approaches for the estimation of large-scale discrete choice models, and finally, dynamic discrete choice models.

### 2.1.1 Decision Rules

The random utility maximization (RUM) framework (McFadden, 1978) is the most widely used approach to model discrete choice behavior. The framework assumes that each individual \( n \) associates an utility \( u_{ni} \) with each alternative \( i \) in a choice set \( C_n \). This utility consists of two parts: a deterministic part \( v_{ni} \) that contains observed attributes, and an random term \( \epsilon_{ni} \) which is unknown to the analyst. The additive RUM framework (Fosgerau et al., 2013b, McFadden, 1978) assumes that the utility can be written as a sum of the two parts \( u_{ni} = v_{ni} + \epsilon_{ni} \). The deterministic term \( v_{ni} \) can include attributes of the alternative as well as socio-economic characteristics of the individual. In general, a linear-in-parameters formula is used, i.e., \( v_{ni} = \beta^T x_{ni} \), where \( T \) is the transpose operator, \( \beta \) is a vector of parameters to be estimated and \( x_{ni} \) is the vector of attributes of alternative \( i \) as observed by individual \( n \). The decision maker aims to maximize the utility, so the choice probability that an alternative \( i \) is chosen by individual \( n \) is

\[
P^{\text{RUM}}(i|C_n) = P(u_{ni} \geq u_{nj}, \forall j \in C_n) = P(v_{ni} + \epsilon_{ni} \geq v_{nj} + \epsilon_{nj}, \forall j \in C_n).
\]

Besides the additive RUM, the multiplicative RUM framework has been proposed by assuming that the summation of \( v_{ni} \) and \( \epsilon_{ni} \) is replaced by multiplication, i.e., \( u_{ni} = v_{ni} \epsilon_{ni} \) (Fosgerau and Bierlaire, 2009). In this context, the choice probability can be derived as \( P^{\text{RUM}}(i|C_n) = P(v_{ni} \epsilon_{ni} \geq v_{nj} \epsilon_{nj}, \forall j \in C_n) \). Different assumptions for the random terms \( \epsilon_{nj}, \forall j \in C_n \), lead to different types of discrete choice models.

Chorus (2010) propose a random regret minimization (RRM) decision rule as an alternative to the RUM framework. The RRM framework is based on the assumption that when decision
makers choose between alternatives, they try to avoid the situation where a non-chosen alternative outperforms a chosen one in terms of the observed attributes. Since its introduction in 2010, the RRM framework has been extended (Chorus, 2012, 2014) and used by a growing number of choice modelers in a variety contexts, for instance de Bekker-Grob and Chorus (2013) apply the RRM for health care, and Hensher et al. (2013) analyze the use of this framework for automobile fuel choice. The RRM decision rule can be translated into a random regret function $r_{ni}$ of alternative $i$ observed by individual $n$. It is the sum of a systematic part and a random error term $\epsilon_{ni}$

$$r_{ni} = \sum_{j \neq i, j \in C_n} \sum_t \ln \left(1 + e^{\beta_t(x_{nj}(t) - x_{ni}(t))}\right) + \epsilon_{ni},$$

where $t$ is an attribute index and $x_{ni}$ is a vector of attributes with respect to individual $n$ and alternative $i$. Contrary to the RUM-based models, a decision maker aims to minimize the random regret, i.e., an alternative is chosen by taking the minimum of $r_{ni}$ over choice set $C_n$. Accordingly, the probability that alternative $i$ is chosen under the RRM framework is

$$P^\text{RRM}(i|C_n) = P(r_{ni} \leq r_{nj}, \forall j \in C_n).$$

A disadvantage of the RRM models, as highlighted in Chorus (2012), is that the running time for computing the choice probabilities increases quadratically as the choice sets become larger since every alternative is compared with every other in terms of each attribute. This leads to the fact that the estimation of the RRM models is more costly, compared to standard RUM-based models, especially with large choice sets.

We present in the following different discrete choice models based on the conventional RUM framework but note that the corresponding RRM-based models can be derived in similar ways.

### 2.1.2 Multinomial Logit

The multinomial logit (MNL) model results from the assumption that the random terms $\epsilon_{ni}$ in (2.1) are independent and identically distributed (i.i.d.) extreme value type I. The corresponding choice probability is

$$P(i|C_n) = \frac{\exp \left(\frac{1}{\mu} v_{ni}\right)}{\sum_{j \in C_n} \exp \left(\frac{1}{\mu} v_{nj}\right)},$$

where $\mu > 0$ is a scale parameter of the extreme value distribution. Due to the simplicity of the choice probabilities, the MNL is widely used in many applications.

The MNL model exhibits the Independence from Irrelevant Alternatives (IIA) property, which implies proportional substitution across alternatives. This property means that for two alternatives, the ratio of the choice probabilities is the same no matter what other alternatives are
available or what the attributes of the other alternatives are. However, if alternatives share unobserved attributes (i.e. random terms are correlated), then the IIA property does not hold. Several models have been developed to relax the IIA property and hence capture the correlation between the random terms. Some well-known examples are the nested (Ben-Akiva, 1973) and cross-nested (Vovsha and Bekhor, 1998) belonging to the multivariate extreme value model family that we present in the following.

2.1.3 Multivariate Extreme Value Models

In order to relax the IIA property from the MNL model, other models have been proposed by making different assumptions on the random terms, e.g., the nested logit (Ben-Akiva and Lerman, 1985, Ben-Akiva, 1973), the cross-nested logit (Vovsha and Bekhor, 1998), the paired comparison logit (Koppelman and Wen, 2000), the generalized nested logit (Wen and Koppelman, 2001), the ordered generalized extreme value (GEV) (Small, 1987), the GenL (Swait, 2001), and specialized compound GEV models (Bhat, 1998, Whelan et al., 2002). These models all belong to the multivariate extreme value (MEV) family of models.

The theory of the MEV model was initially introduced by McFadden (1978). It is assumed that the joint distribution of the random terms \( (\epsilon_1, \ldots, \epsilon_J) \), where \( J \) is the number of alternatives in the choice set, is the MEV distribution with cumulative distribution function

\[
F(x_1, \ldots, x_J) = e^{-G(e^{-x_1}, \ldots, e^{-x_J})},
\]

(2.3)

where \( G : \mathbb{R}^J \to \mathbb{R} \) is the choice probability generating function (CPGF) of the MEV model (Fosgerau et al., 2013b). In order to be consistent with the McFadden’s RUM theory, this function needs to satisfy four conditions

1. \( G(.) \) is non negative,

2. \( G(.) \) is homogeneous of degree \( \mu > 0 \) that is \( G(\lambda z) = \lambda^\mu G(z) \),

3. \( \lim_{z_r \to \infty} G(z_1, \ldots, z_J) = \infty \), \( \forall r \) such that \( 1 \leq r \leq J \),

4. The mixed partial derivatives of \( G(z_1, \ldots, z_J) \) exist and are continuous. Moreover, the \( k^{th} \) partial derivative with respect to component \( z_i \) is non negative if \( k \) is odd, and non-positive if \( k \) is even.

The probability of choosing alternative \( i \) within the choice set \( C_n \) of a given decision maker \( n \) is

\[
P(i|C_n) = \frac{z_i \frac{\partial G}{\partial z_i}(z_1, \ldots, z_J)}{\mu G(z_1, \ldots, z_J)},
\]

(2.4)
where \( z_i = e^{v_{ni}} \). We note that that the original formulation proposed by McFadden (1978) is based on \( \mu = 1 \). The generalization with \( \mu > 0 \) was derived by Ben-Akiva and François (1983). The MNL model is an instance of the MEV model where the corresponding CPGF is \( G(z_1, \ldots, z_J) = \sum_{i=1}^{J} z_i^\mu \). The MEV model provides an appealing framework to develop closed forms for RUM models.

The four conditions for the CPGF can be difficult to verify. Daly and Bierlaire (2006) show how to define CPGFs for a group of MEV models by a rooted, cycle-free graph with certain characteristics on the parameters associated with nodes and arcs. This graph is referred to as a MEV-network (also known as GEV-network). This way of defining a MEV model is useful because it is just necessary to verify that the graph is a MEV-network instead of verifying that the CPGF satisfies the MEV conditions. The network MEV model generalizes many MEV models in the literature, e.g., the MNL, nested logit, cross-nested logit models. In this context, we note that the cross-nested model is flexible, since it can approximate any additive RUM model (Fosgerau et al., 2013b).

### 2.1.4 Mixed Multinomial Logit

The mixed MNL (MMNL) model is convenient to relax the IIA property of the MNL model. The model is fully flexible in the sense that it can approximate any random utility model (McFadden and Train, 2000). The model has been widely used in practice due to this convenience. The first application of the MMNL was the demand for electricity-using goods (Electric Power Research Institute, 1977).

In the MMNL model, the probabilities are the integrals of standard logit probabilities over a density \( f(\beta) \) of parameters \( \beta \)

\[
P(i|C_n) = \int L(i|C_n) f(\beta) d\beta,
\]

where \( L(i|C_n) \) is the logit probability evaluated at parameters \( \beta \). Consequently, the form of the choice probability is

\[
P(i|C_n) = \frac{\exp \left( \frac{1}{\mu} v_{ni} \right)}{\sum_{j \in C_n} \exp \left( \frac{1}{\mu} v_{nj} \right)} f(\beta) d\beta.
\]

In the case of panel data (more than one observation for an individual), the correlation can be captured by assuming that the parameters \( \beta \) do not vary for the same individual, while being randomly distributed throughout the population (Revelt and Train, 1998). In this case, the
choice probability has the following form

\[ P(i|C_n) = \int \prod_{t=1}^{T_n} \frac{\exp\left(\frac{1}{\mu} v_{nti}\right)}{\sum_{j \in C_n} \exp\left(\frac{1}{\mu} v_{ntj}\right)} f(\beta) d\beta, \]

where \( T_n \) (\( T_n > 1 \)) is the number of observations available for individual \( n \), and \( i_t \) is the \( t^{th} \) observed choice.

The choice probability of the MMNL model can be derived in two different ways. The first way is based on random coefficients where the parameters \( \beta \) of the utility \( v_{ni} \) vary over decision makers in the population with density \( f(\beta) \). The second way adds error components with zero mean to the utilities in order to create correlation among the utilities for different alternatives. Although these approaches are formally equivalent, they provide different interpretations.

The estimation of the MMNL model implicates an integration over the distribution of the random parameters. The integration can be approximated numerically by sampling over the random parameters. The sample can be generated by standard Monte Carlo or quasi-Monte Carlo techniques. Randomized quasi-Monte Carlo methods have also been considered for mixed logit models, while the improvement is sometimes limited (Munger et al., 2012).

### 2.1.5 Sampling of Alternatives

When the choice sets are very large, model estimation can become expensive. In this context, sampling of alternatives becomes appealing. The idea is to sample alternatives from the full choice sets, and correct the choice probabilities so that the parameter estimates are consistent. McFadden (1978) demonstrated that the MNL model can be consistently estimated under sampling of alternatives by simply adding corrections to the choice probabilities. Guevara and Ben-Akiva (2013a), Guevara and Ben-Akiva (2013b) and Guevara et al. (2014) recently showed how to obtain consistent estimates under sampling of alternatives for the MEV, mixed logit and RRM-based logit models. We briefly present these approaches in the following.

McFadden (1978) show that the MNL model can be consistently estimated with sampled choice sets by using the following corrected choice probabilities

\[ P(i|D_n) = \frac{\exp\left(\frac{1}{\mu} v_{ni} + \ln q(D_n|i)\right)}{\sum_{j \in D_n} \exp\left(\frac{1}{\mu} v_{nj} + \ln q(D_n|j)\right)}, \]

where \( D_n \) is a sampled choice set, and \( q(D_n|i) \) is the probability of sampled choice set \( D_n \) given that \( i \) is the chosen alternative. For the MEV model with CPGF \( G(\cdot) \), the choice probability
can be corrected as (Bierlaire et al., 2008)

\[ P(i|D_n) = \frac{\exp (v_{ni} + \ln G_i(e^{v_{nt}}, t \in C_n) + \ln q(D_n|i))}{\sum_{j \in D_n} \exp (v_{nj} + \ln G_j(e^{v_{nt}}, t \in C_n) + \ln q(D_n|j))}, \]

where \( G_i(e^{v_{nt}}, t \in C_n) \) is the partial derivative of \( G \) with respect to \( e^{v_{nt}} \). This formula is however impractical since the terms \( \ln G_i(e^{v_{nt}}, t \in C_n), i \in C_n \), require the calculation over the full choice sets. An approximation of \( G_i \) based on a sampled choice set is proposed by Guevara and Ben-Akiva (2013a). This approximation requires additional sampled choice sets, and the corrected choice probability becomes

\[ P(i|D_n) = \frac{\exp \left( v_{ni} + \ln \tilde{G}_i(e^{v_{nt}}, t \in D'_n) + \ln q(D_n|i) \right)}{\sum_{j \in D_n} \exp \left( v_{nj} + \ln \tilde{G}_j(e^{v_{nt}}, t \in D'_n) + \ln q(D_n|j) \right)}, \]

where \( D'_n \) is another sampled choice set for the approximation of \( G_i \), and \( \tilde{G}_i \) is the corresponding approximated function of \( G_i \).

Guevara and Ben-Akiva (2013b) also show that the mixed logit model can be consistently estimated under sampling of alternatives by using the following corrected choice probability

\[ P(i|C_n) = \int W_n \frac{\exp \left( \frac{1}{\mu} v_{ni} \right)}{\sum_{j \in C_n} \exp \left( \frac{1}{\mu} v_{nj} \right)} f(\beta) d\beta, \]

where the term \( W_n \) is constructed using alternatives in \( D_n \) (see for instance Guevara and Ben-Akiva, 2013b). Finally, the reader can consult Guevara et al. (2014) for the importance sampling approach for RRM-based logit models.

2.1.6 Dynamic Discrete Choice Models

In some situations, decision makers make sequential choices, and each time with an eye towards how the decision will impact the future outcome. Dynamic discrete choice models can be used to analyze this type of sequential looking-forward choice behavior. These models are also known as parametric Markov decision processes. Dynamic discrete choice models have been used in various applications, e.g., Wolpin (1984) on fertility and child mortality, Miller (1984) on job matching and occupational choice, and Rust (1987) on bus engine replacement. We focus this review on the framework proposed by Rust (1987) which is relevant to the models and methods proposed in this thesis. The description of the dynamic discrete choice framework is based on Aguirregabiria and Mira (2010), and we refer the reader to this paper for more details.
We assume that time is discrete and indexed by $t$, $t = 1, \ldots, T$, where the time horizon $T$ can be either finite or infinite. At each time period $t$, an individual $n$ observes a vector of state variables $s_{nt}$ and chooses an alternative $a_{nt}$ (also known as action) from a discrete and finite set $\mathcal{H} = \{1, \ldots, J\}$. The utility associated with individual $n$ over possible sequences of states is represented by

$$
\sum_{j=0}^{T} \rho^j u(a_{n,t+j}, s_{n,t+j}),
$$

where $\rho \in (0, 1)$ is a discount factor and $u(a_{n,t+j}, s_{n,t+j})$ is the utility function at time $t + j$ associated with alternative $a_{n,t+j}$ and state $s_{n,t+j}$. At period $t$, the individual selects an alternative $a_{nt} \in \mathcal{H}(s_{nt})$ that maximizes the expected utility

$$
E \left( \sum_{j=0}^{T} \rho^j u(a_{n,t+j}, s_{n,t+j} | a_{nt}, s_{nt}) \right),
$$

where $\mathcal{H}(s_{nt})$ is the set of available alternatives at state $s_{nt}$. This can be formulated as a dynamic programming problem. If we denote by $V(s)$, for all states $s$, the expected maximum utilities, or the value functions, of the dynamic programming problem over all the stochastic components of the model. These values are recursively defined by Bellman’s equation

$$
V(s_{nt}) = \mathbb{E} \left[ \max_{a \in \mathcal{H}(s_{nt})} \left\{ u(a, s_{nt}) + \rho \int V(s_{n,t+1}) dF(s_{n,t+1} | a, s_{nt}) \right\} \right],
$$

where $F(\cdot)$ is a Markov transition distribution function, which refers to the individual’s beliefs about future states. Now we assume that the vector of state variables $s_{nt}$ can be separated into two subsets of state variables: $s_{nt} = (x_{nt}; \epsilon_{nt})$, where sub-vector $x_{nt}$ contains observed attributes, and $\epsilon_{nt}$ is unknown to the analyst. In this setting, Rust (1987) proposed a simple framework which could be described as a dynamic logit model. Under this framework, the utility $u(a, s_{nt})$ is expressed as a sum of observable $v(a, x_{nt})$ and unobservable $\epsilon_{nt}(a)$ components, i.e., $u(a, s_{nt}) = v(a, x_{nt}) + \epsilon_{nt}(a)$. The terms $\{\epsilon_{nt}(a), a = 1, \ldots, J\}$ are assumed to be i.i.d extreme value type I with zero mean, and they are independent of everything in the model. This also refers to the so-called CLOGIT assumption. Moreover, the support set of $x_{nt}$ is assumed to be discrete and finite, i.e., $x_{nt} \in \mathcal{X} = \{x^1, \ldots, x^{|\mathcal{X}|}\}$, where $|\mathcal{X}|$ is finite. Under these assumptions, the expected maximum utility over the distribution of unobservable terms, conditional on the observable terms, is a solution to the following Bellman’s equation

$$
V(x_{nt}) = \mathbb{E}_x \left[ \max_{a \in \mathcal{H}(s_{nt})} \left\{ v(a, x_{nt}) + \epsilon_{nt}(a) + \rho \sum_{x_{n,t+1} \in \mathcal{X}} V(x_{n,t+1}) f_{\mathcal{X}}(x_{n,t+1} | a, x_{nt}) \right\} \right], \quad (2.5)
$$

where $f_{\mathcal{X}}(\cdot)$ is the Markov transition function defined based on the support of the states. Under the CLOGIT assumption, the model at each stage is MNL, so the expected maximum utility in
(2.5) is given by the logsum

\[ V(x_{nt}) = \ln \left( \sum_{a \in H(x_{nt})} \exp \left( v(a, x_{nt}) + \rho \sum_{x_{n,t+1} \in X} V(x_{n,t+1}) f_X(x_{n,t+1} | a, x_{nt}) \right) \right), \]

and the choice probabilities are

\[ P(a|x_{nt}) = \frac{\exp \left( v(a, x_{nt}) + \rho \sum_{x_{n,t+1} \in X} V(x_{n,t+1}) f_X(x_{n,t+1} | a, x_{nt}) \right)}{\exp (V(x_{nt}))}. \]

We are interested in estimating the structural parameters, transition probabilities, and the discount factor \( \rho \). This can be done by using an estimation criterion such as the maximum likelihood. Suppose that we have a panel data of \( N \) individuals who make sequential decisions and behave according to a dynamic discrete choice process. For an individual \( n \), at time period \( t \), state \( x_{nt} \) and the corresponding chosen alternative \( a_{nt} \) are observed. The data set is therefore: \( \text{Data} = \{(a_{nt}, x_{nt}), \ n = 1, \ldots, N; \ t = 1, \ldots, T_n \} \), where \( T_n \) is the number of time periods over which the decisions of individual \( n \) are observed. The log-likelihood can be defined based on the data set as

\[ LL(\beta) = \sum_{n=1}^{N} \ln \left( \Pr ( (a_{nt}, x_{nt}), \ t = 1, \ldots, T_n | \beta) \right), \]

where \( \beta \) is the vector of the structural parameters to be estimated. In order to evaluate the log-likelihood function for a given \( \beta \), it is necessary to solve the dynamic programming problem exactly or approximately. Rust (1987) proposed the nested fixed point algorithm that can be used to solve the dynamic programming problem, and other algorithms for the estimation of dynamic discrete choice models can be consulted in Aguirregabiria and Mira (2010). We discuss in more detail estimation methods for dynamic discrete choice models in Section 2.2.

It is important to note that CLOGIT is an essential assumption in the Rust’s model in order to make the dynamic models convenient to estimate. This assumption however ignores the correlation between unobserved terms. There are also studies in which the CLOGIT assumption has been relaxed, and we refer the reader to Aguirregabiria and Mira (2010) for details.

2.2 Maximum Likelihood Estimation

Parameters of a discrete choice model can be estimated by maximum likelihood. This is done by defining a log-likelihood function over a sample of observations and maximizing this function using nonlinear optimization techniques. In this section, we first present the definition of the log-likelihood function as well as its properties. We then focus on optimization methods used to maximize this function. Finally, we review dynamic programming methods for estimating dynamic discrete choice models.
2.2.1 Log-Likelihood Function and Properties

In maximum likelihood estimation, we aim at solving the following maximization problem

\[
\max_\beta \hat{L}_N(\beta) = \frac{1}{N} \sum_{n=1}^{N} \ln P(i_n|\beta) \quad (2.6)
\]

where \( P(I|\beta) \) is a mass probability function, defined on random variable \( I \), conditioned on a set of parameters \( \beta \), and \( i_1, \ldots, i_N \) are given observations, corresponding to individuals \( n = 1, \ldots, N \). Using the terminology popular in stochastic programming (SP) and assuming that the individuals are randomly drawn from a population \( I \), (2.6) can be seen as the sample average approximation of the “true” problem

\[
\max_\beta LL(\beta) = \mathbb{E}_I[\ln P(I|\beta)]. \quad (2.7)
\]

We note here that \( P \) does not necessarily correspond to the probability mass of \( I \) over the population, in which case the model is said to be misspecified. We can however still refer to the SP literature to establish that, under some regularity conditions, when \( N \) rises to infinity,

\[
\text{dis}(\hat{S}_N, S^*) \rightarrow 0 \text{ almost surely}, \quad (2.8)
\]

where \( \text{dis} \) is a distance measure, \( \hat{S}_N \) and \( S^* \) are the sets of first-order critical points of (2.6) and (2.7), respectively, assuming that \( \hat{S}_N \) and \( S^* \) are not empty (see e.g. Shapiro, 2003 and Shapiro et al., 2009, Chapter 5). Moreover, if these sets are singletons, we denote by \( \hat{\beta}_N \) the solution of (2.6), and by \( \beta^* \) the solution of (2.7). (2.8) indicates that \( \hat{\beta}_N \rightarrow \beta^* \) almost surely when \( N \) rises to infinity. This property is referred to as the consistency of the maximum likelihood estimator. Moreover, we also have

\[
\sqrt{N}(\hat{\beta}_N - \beta^*) \Rightarrow \mathbb{N}(0, \Psi),
\]

where \( \Rightarrow \) designs the convergence in distribution, and \( \mathbb{N} \) refers to the normal distribution. Setting the gradient of (2.6) to zero, it can be shown that (Newey and McFadden, 1994, White, 1982)

\[
\Psi = H(\beta^*)^{-1}I(\beta^*)H(\beta^*)^{-1},
\]

where \( H(\beta^*) = \mathbb{E}_I[\nabla_\beta P(I|\beta^*)] \) and \( I(\beta^*) = \mathbb{E}_I[\nabla_\beta P(I|\beta^*)\nabla_\beta f(I|\beta^*)^T] \), where \( T \) is the transpose operator, is the outer product of scores, also called the Fisher’s information matrix. The asymptotic variance-covariance can therefore be estimated using

\[
\text{Cov}(\hat{\beta}_N) = \frac{[H_N(\hat{\beta}_N)]^{-1}I_N(\hat{\beta}_N)[H_N(\hat{\beta}_N)]^{-1}}{N}, \quad (2.9)
\]
where
\[ H_N(\hat{\beta}_N) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\beta^2 \ln P(i_n|\hat{\beta}_N) \]
and
\[ I_N(\hat{\beta}_N) = \frac{1}{N} \sum_{n=1}^{N} [\nabla_\beta \ln P(i_n|\hat{\beta}_N)][\nabla_\beta \ln P(i_n|\hat{\beta}_N)]^T \]
are the samples average estimates of the Hessian and the information matrix, respectively. We refer to the variance-covariance matrix given by (2.9) as the robust variance-covariance matrix.

The well-known information matrix equality implies that if the model is well specified, i.e. \( P(\mathcal{I}|\beta^*) \) is the mass probability of \( \mathcal{I} \) over the population, the Fisher information matrix is equal to the opposite of the Hessian matrix, i.e., \( I(\beta^*) = -H(\beta^*) \). The robust variance-covariance of the maximum likelihood estimator then becomes \( -[H_N(\hat{\beta}_N)]^{-1}/N \). In this context, we note that, based on the information matrix equality, White (1982) proposes a statistical test to verify whether a probabilistic model is correctly specified (see Section 2.3 for more details).

### 2.2.2 Maximizing the Log-Likelihood Function

The maximum likelihood estimation problem (2.7) can be viewed as an unconstrained nonlinear optimization problem “\( \min_{x \in \mathbb{R}^d} f(x) \)”, where \( f(x) \) is the opposite of the log-likelihood function. Most nonlinear optimization algorithms use the value of the objective function and possibly its first and second derivatives (Hessian), under the assumption that \( f \) is twice continuously differentiable. More specifically, Newton methods require the exact Hessian to be available, but the evaluation of the true Hessian may be computationally costly. Quasi-Newton approaches rely on approximations of the Hessian, with the hope of retaining fast local convergence at a lower cost. In this section, we first review two classes of optimization algorithms that ensure theoretical convergence to a solution from any starting point: line search and trust region methods. Next, we describe popular methods for approximating the Hessian matrix. We refer the reader to Nocedal and Wright (2006) for more details.

At each iteration \( k > 0 \), a line search algorithm computes a search direction \( p_k \) and a positive step length \( \alpha_k \) along the search direction that satisfies some conditions. A popular option in the context is the Wolfe conditions, which can be expressed as

\[
\begin{align*}
    f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k \\
    |\nabla f(x_k + \alpha_k p_k)^T p_k| &\geq c_2 |\nabla f(x_k)^T p_k|,
\end{align*}
\]

(2.10)

where \( x_k \) is the solution at iteration \( k \), \( p_k \) is a search direction, \( c_1, c_2 \) are two constants such that \( c_1 \leq c_2 < 1 \), and \( \nabla f(\cdot) \) is the gradient of \( f(\cdot) \). Most line search algorithms require \( p_k \) to be a descent direction, i.e., \( p_k^T \nabla f(x_k) < 0 \), thus reducing the function \( f(x) \) along this direction.
A trivial option is $p_k = -\nabla f(x_k)$, which refers to the steepest descent method. However, this approach often leads to poor rate of convergence. Alternatively, $p_k$ can be computed as the Newton direction $-H_k^{-1}\nabla f(x_k)$, where $H_k$ is the Hessian or an approximation of the Hessian at $x_k$, assuming that $H_k$ is non-singular. The Newton direction is a descent direction as long as $H_k$ is positive definite, otherwise a positive definite matrix can be constructed as $H_k + \nu I$ where $I$ is the identity matrix and $\nu$ is a sufficiently large positive scalar. Given a decent direction, if a scalar $\alpha_k$ provides a successful iterate, i.e., it satisfies the Wolfe conditions, then $x_{k+1}$ is set to $x_k + \alpha_k p_k$ and the algorithm goes to next iteration.

The trust region method is another popular optimization technique. The basic idea is, at iteration $k$, to minimize an approximate model of the objective function within a region centered at the current iterate $x_k$:

$$\min_{s \in B_k(x_k)} m_k(s).$$  \hspace{1cm} (2.11)

Solving (2.11) has to be cheap, compared to the original optimization problem, and a quadratic model is typically used, i.e.

$$m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T H_k s.$$  

(2.11) can be solved approximately only, and the region $B_k$ prevents to produce too long steps as the model reflects the true objective in a neihborhood of $x_k$ only. $B_k$ is called the trust region, and it is usually defined as a sphere centered at $x_k$, i.e. $B_k = \{s \in \mathbb{R}^\ell \text{ and } ||s|| \leq \Delta_k\}$, where $\Delta_k$ is the trust region radius. Let $s_k$ be the solution of (2.11). A candidate is computed as $x_k + s_k$, and the predicted reduction obtained at $m_k(s_k)$ is compared with the true objective function reduction by means of the ratio

$$\varrho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}.$$  

If the agreement $\varrho_k$ is sufficiently good, the trial point $x_k + s_k$ becomes the new iterate and the trust region is maintained or enlarged if $\varrho_k$ is close to one (or even bigger than one). Otherwise, the trust region is shrunk in order to improve the quality of the model. We refer the reader to Conn et al. (2000a) for a comprehensive coverage of trust region methods.

Line search, based on the (quasi-)Newton direction, and usual trust region methods make intensive use of the quadratic model $m_k$. As mentioned earlier, an approximation of the Hessian is often used instead of the true Hessian due to its high computational cost. In the following we describe some well-know approaches for approximating the Hessian matrix.

The BHHH (also known as the statistical approximation) (Berndt et al., 1974) is popularly used for the maximum likelihood estimation due to the information matrix equality. Berndt et al. (1974) suggested to approximate the Hessian with the sample average estimation of the
information matrix

\[ H_{BHHH} = - \frac{1}{N} \sum_{n=1}^{N} \nabla \ln P(i_n|\beta) \nabla \ln P(i_n|\beta)^T. \]

This approximation only requires the gradients, so it is useful when dealing with a complex log-likelihood function.

We note that, in smooth optimization problems, the secant methods are widely used. The idea is that at each iteration \( k \) the curvature information is used to approximate the Hessian matrix. More precisely, we aim to find a Hessian approximation \( H_k \) that satisfies the secant equation \( H_k s_k = y_k \), where \( s_k = x_{k+1} - x_k \) and \( y_k = \nabla f(x_k) - \nabla f(x_{k-1}) \). A well-known approximation based on this equation is the BFGS method where the Hessian approximation is updated as

\[ H_k = H_{k-1} - \frac{H_{k-1} s_k s_k^T H_{k-1}}{s_k^T H_{k-1} s_k} + \frac{y_k y_k^T}{y_k s_k}, \]

where \( H_{k-1} \) is the Hessian approximation at iteration \( k - 1 \). This update is also called the rank-2 update. The SR1 or rank-1 update is another secant method which is also popular in the context. Using the SR1 method, the Hessian can be approximated as

\[ H_k = H_{k-1} + \frac{(y_k - H_{k-1} s_k)(y_k - H_{k-1} s_k)^T}{(y_k - H_{k-1} s_k)^T s_k}. \]

Both the BFGS and SR1 maintain the symmetry of the matrix. The BFGS guarantees positive definiteness while the SR1 method does not, so it allows to take the advantage of a negative curvature (Conn et al., 1991).

The BFGS and SR1 methods are easy to apply since they require only the first derivatives of the log-likelihood function. The convergence of Hessian approximations to the true Hessian may, however, be slow, resulting in numerous iterations during the optimization process. On the opposite, the BHHH update better reflects the problem structure due to the information matrix equality. However, the BHHH is based on the assumption that the information matrix equality holds, and that the number of observations is large enough. In many contexts, this is not true, and the optimization algorithm using BHHH may fail to converge (Mai et al., 2015d). The BHHH method is however similar to the Gauss-Newton technique for nonlinear least-squares, opening the possibility to develop structured quasi-Newton approaches. While they have been extensively studied for least-squares problems (e.g. Al-Baali and Fletcher, 1986, Dennis Jr et al., 1981, Toint, 1987), few studies for maximum likelihood estimation exist, and only with the trust region method (Bunch, 1987, 1988). We will explore these questions in more details in Chapter 9.
2.2.3 Estimating Dynamic Discrete Choice Models

The estimation of dynamic discrete choice models is considerably more difficult than static models since it requires the use of algorithms in which the value functions are solved in the optimization of the estimation criterion, e.g., maximum likelihood. In this section, we introduce the estimation methods used to estimate the Rust’s model. Other estimation methods for dynamic discrete choice models can be consulted in Aguirregabiria and Mira (2010).

Rust (1987) proposed the nested fixed point algorithm (NFXP) to compute the maximum likelihood estimates of his structural dynamic discrete choice models. Formally, the NFXP can be formulated as a constrained optimization problem

$$\max_{\beta, V} LL(\beta, V) \text{ subject to } V = T_\beta(V),$$

(2.14)

where $\beta$ is the vector of structural parameters, $V$ is the value function, and $V = T_\beta(V)$ is a contraction mapping defined based on Bellman’s equation. One can show that $||T_\beta(V) - T_\beta(V')|| \leq \varsigma ||V - V'||$, where $\varsigma \in [0, 1)$. It is guaranteed that $V = T_\beta(V)$ has a unique fixed point solution $V_\beta$ that can be computed by the method of successive approximations. So the constrained optimization problem in (2.14) reduces to

$$\max_{\beta, V_\beta} LL(\beta, V_\beta).$$

It means that one has to compute the fixed point solution in order to evaluate the log-likelihood function $LL(\beta)$. The NFXP algorithm consists of an outer iterative nonlinear optimization algorithm for searching over the parameter space and an inner algorithm for computing the fixed point solution. The outer optimization algorithm is based on a nonlinear optimization algorithm, e.g., the trust region or line search methods presented in the previous section, and the inner fixed point algorithm is the method of successive approximations to compute the fixed point solution (value functions) for a given vector of structural parameters. The method of successive approximations can be done as follows. It starts with an initial guess for the fixed point $V^0$, and approximates the fixed point solution by iteratively performing $V^{k+1} = T_\beta(V^k)$. Since one can show that $V^k \to V_\beta$ as $k \to \infty$, the method stops when the fixed point solution is found. The method of successive approximations is also known as the “contraction iterations”.

Rust (1988) also proposed the Newton-Kantorovich method, an alternative to the contraction iterations. The idea is to convert the problem $V = T_\beta(V)$ into the problem of finding $F(V) = V - T_\beta(V) = 0$. Using a standard Taylor series expansion of equation $F(V^{k+1}) = 0$, one can derive a Newton-Kantorovich iteration as

$$0 = F(V^{k+1}) \approx F(V^{k}) + \nabla_{V} F(V^{k})(V^{k+1} - V^{k}),$$
or equivalently,

\[ V^{k+1} = V^k + \left[ I - \nabla_V T_\beta(V^k) \right]^{-1} \left[ V^k - T_\beta(V^k) \right], \tag{2.15} \]

where \( I \) is the identity matrix. The Newton-Kantorovich iteration requires deriving the first derivative of \( T_\beta \) with respective to \( V \), and solving a system of linear equations. We note that the invertibility of \( [I - \nabla_V T_\beta(V^k)] \) is guaranteed (Rust, 1988).

The Newton-Kantorovich iterations are more difficult to perform, compared to the contraction iterations, since they require the first derivative of the contraction mapping. However, the contraction iterations become slow when \( V^k \) is close to the fixed point solution, especially when the discount factor is close to 1. On the contrary, the Newton-Kantorovich iterations typically converge rapidly from any estimate \( V^k \) which is in a sufficiently small neighborhood of the solution \( V_\beta \). This has led Rust (1988) to propose the polyalgorithm which is based on a combination of the Newton-Kantorovich and contraction iterations.

Several alternatives to the NFXP have also been proposed in the econometric literature. Hotz and Miller (1993) observe that it is not necessary to completely solve the fixed point problem in order to estimate the model parameters, and develop the conditional choice probability (CCP) method that exploits the mappings from the value functions to the choice probabilities. Aguirregabiria and Mira (2002) propose the nested pseudo-likelihood algorithm (NPL), where the idea is to swap the order of the outer and inner algorithms so that the outer algorithm solves the value functions and the inner algorithm maximizes the pseudo-likelihood function. This approach is useful if the value functions are costly to evaluate. More recently, Su and Judd (2012) propose a constrained optimization approach for the maximum likelihood estimation of infinite horizon dynamic discrete choice models, called the mathematical programming with equilibrium constraints (MPEC). They argue that the MPEC outperforms the NFXP with contraction iterations in terms of computational time. However, Iskhakov et al. (2015) recently showed that when combining contraction and Newton-Kantorovich iterations to solve the fixed point problem, the NFXP has a similar performance to the MPEC in terms of speed.

### 2.3 Statistical Tests of Model Misspecification

Tests for model misspecification are used to statistically determine whether a model correctly describes choices in the population from which the estimation data were sampled. While there are many different tests in the literature, we focus this review on those relevant to the work presented in this thesis.

Hausman and McFadden (1984) propose two different approaches to verify whether the IIA property holds for a MNL model. The first approach is based on a nested logit as an alternative model, and one can apply classical test procedures such as the Wald, likelihood ratio test or
Lagrange multiplier test (Engle, 1984). However, this test is based on the assumption that the alternative specification on which the tests are based is correct, which may not be satisfied. The second approach does not require an alternative model. Instead, it is based on estimation results of the model using the full choice sets and sub-choice sets of alternatives. If we denote $\hat{\beta}_c$ and $\Omega_c$ be the parameter estimates and variance-covariance matrix obtained by all the observations and full choices set $C_n$, and $\beta_s$ and $\Omega_s$ are those obtained by a subsample and sub-choice sets $S_n$, then the following asymptotic chi-square distribution is used for the IIA test

$$(\hat{\beta}_c - \hat{\beta}_s)(\Omega_c - \Omega_s)^{-1}(\hat{\beta}_c - \hat{\beta}_s) \Rightarrow \chi^2_{\eta},$$

where $\eta$ is the degree of the chi-square distribution which is equal to the rank of $(\Omega_c - \Omega_s)$ under the null hypothesis. One can compute the $p$-values based on the $\chi^2_\eta$ distribution and reject or accept the null hypothesis, i.e., if the $p$-value does not exceed the critical value, the null hypothesis that the IIA property holds is rejected, otherwise we cannot reject the null hypothesis.

The Lagrange multiplier test proposed by McFadden and Train (2000) is another test for the MNL model. This allows to test whether mixing is needed by solely using MNL model estimation results. The test is based on constructing artificial variables, and using a Wald or likelihood ratio test for the hypothesis that the artificial variables should be omitted from the MNL model. McFadden and Train (2000) show that the test is asymptotically equivalent to a Lagrange multiplier test of the hypothesis of no mixing against the alternative of a mixed MNL model.

White (1982) proposes a general test for probabilistic models based on the information matrix identity (for instance Newey and McFadden, 1994), called the information matrix test. This test is based on the jointly asymptotically distributed Normal property of the sum $D(\hat{\beta}_N) = I_N(\hat{\beta}_N) + H_N(\hat{\beta}_N)$, where $\hat{\beta}_N$ is the maximum likelihood estimator, $N$ is the number of observations, and $I_N(\hat{\beta}_N)$, $H_N(\hat{\beta}_N)$ are the sample estimators of the information matrix and the true Hessian. White (1982) shows that, under the null hypothesis that the model is correctly specified,

$$\sqrt{N}D(\hat{\beta}_N) \Rightarrow N(0, V(\hat{\beta}_N)),$$

where $V(\hat{\beta}_N)$ is the variance-covariance matrix of the test. Therefore, an asymptotic $\chi^2$ statistic test can be derived as

$$D(\hat{\beta}_N)^T V(\hat{\beta}_N)^{-1} D(\hat{\beta}_N) \Rightarrow \frac{1}{N} \chi^2_{\eta},$$

(2.16)

where $\eta$ is the number of indicators of interest in $I$ and $H$. The information matrix test then can be done by computing the $p$-value using (2.16). This test does not require the estimation of an alternative model, and can be applied to a wide range of discrete choice models. However, the computation of $V(\hat{\beta}_N)$ requires the third derivative of the log-likelihood function, which makes the test difficult to perform. Different alternatives to White’s test have been proposed to simplify
the test, e.g., Chesher (1983) and Lancaster (1984) show how to compute the information matrix test without the need of third derivatives. Orme (1990) shows that the different variants of the information matrix test statistic can be improved by including expected values of third order derivatives. Horowitz (1994) shows how to use the bootstrap technique to overcome the problem of incorrect finite-sample size of the information matrix test. He also notes that the bootstrap method is computationally expensive, and proposes a cheaper variant to obtain finite sample critical values and compares those with the asymptotic critical values.

Finally, we note that misspecification tests can also be used to determine whether a model estimated using a sample from one population correctly describes choices in holdout samples (Horowitz and Louviere, 1993, Louviere and Johnson, 1990).

2.4 Route Choice Analysis using Discrete Choice Models

In this section we focus on the route choice modeling where the purpose is to determine which route a traveler would choose for a given origin destination pair and transport mode (e.g. car or bike). In this context, it is possible to use shortest path algorithms that can identify the shortest path with respect to a link-additive generalized cost function. While these algorithms allow to compute shortest paths very fast, even for large networks, it is not realistic to assume that the analyst has perfect knowledge of travelers’ generalized cost functions. This motivates the use of discrete choice models that assigns probabilities to a set of paths. The resulting discrete distribution can be used to simulate choices, e.g., in a traffic simulation context. Route choice models can be used to assess travelers’ preferences regarding characteristics of different routes (e.g. travel time, travel cost, number of crossings), considering the characteristics of travelers, e.g., age, gender, and trip purpose. In this section we present some models and methods that have been designed specifically for the route choice problem.

Route choice analysis using discrete choice models has some special characteristics that makes it complicated to deal with. First, choice sets of paths are unknown to the analyst and the sets of all feasible paths between an origin-destination pair (also known as universal choice sets) cannot be enumerated. Second, path utilities may be highly correlated, for instance, due to physical overlap in the network. We discuss these issues in some more depth in the following.

We can identify two different views on choice set definition in the literature as those that limit the models to a subsets of alternatives (so called consideration sets) and those that use the universal choice sets. Most studies aim at defining consideration sets (e.g. Ben-Akiva and Bierlaire, 1999a, Ben-Akiva et al., 1984, Frejinger and Bierlaire, 2007). In this case, the analyst uses an algorithm that defines a set of paths (subset of the universal choice set) and these sets are treated as the actual choice sets (all other paths have a zero choice probability). A large number of
algorithms have been proposed for this purpose, to mention a few, the *link elimination* (Azevedo et al., 1993), *link penalty* (de la Barra et al., 1993), *constrained k-shortest paths* (Vanderbei and Shanno, 1997), *labeling* (Ben-Akiva et al., 1984), *branch-and-bound* (Friedrich et al., 2001), and *doubly stochastic* (Bovy and Fiorenzo-Catalano, 2006). We refer the reader to Prato (2009) for a review. The use of consideration sets is supported by the proposition that consumers do not consider all alternatives when making choices (Ratchford, 1980). This may provide a more realistic representation of the choice process, and may lead to a better explanation of consumer behavior. Studies in marketing literature have observed improved predictions when using consideration sets (Hauser and Gaskin, 1984).

When analyzing route choices in real networks (revealed preferences data), only the chosen paths are observed and the analyst does not have any information on alternatives that the traveler considered. Certain studies use GPS data and a travel diary to get some more information (e.g. de Moraes Ramos, 2015), but the data about alternative paths still remain fairly limited and less detailed than the observed choice. The definition of consideration sets of paths therefore rely on how similar alternatives paths are to the observed paths (coverage measure proposed by Ramming, 2002). We are not aware of any study in the literature that reports a 100% coverage which means that the choice set generation algorithms cannot generate all the observed paths.

Horowitz and Louviere (1995) investigate the role of consideration sets in choice modeling. They argue that choice need not be modeled as a two stage process, i.e., the first stage is creating consideration sets and the second concerns choosing an alternative from the consideration sets. The indicators of consideration sets can be used in the decision maker's utility function because they are also reflections of preferences. Moreover, the use of consideration sets may lead to a misspecified model that would provide erroneous forecasts. In the context of route choice, studies have reported the absence of observations in the consideration choice sets (e.g. Ramming, 2002), and inconsistent parameter estimates (estimates change as the choice set definition changes, we illustrate this in an example in Section 2.4.3). This motivates the use of universal choice sets which is an alternative view on choice set definition.

There are two approaches that allow to base the choice model on universal choice sets. One is path-based and uses a sampling of alternatives approach, and the other is network based (recursive logit proposed by Fosgerau et al., 2013a). In this thesis, we focus on route choice models that are based on universal choice sets and we therefore review them in more detail in the following two sections. We emphasize that many models for route choice with consideration sets have been proposed. For example, Cascetta et al. (1996) propose the C-Logit model by adding the Commonality Factor (CF) attribute to the utilities, and Ben-Akiva and Ramming (1998) and Ben-Akiva and Bierlaire (1999b) propose the Path Size Logit (PSL) model based on the idea of the C-logit where the Path Size (PS) attribute for overlapping paths is added to the deterministic part of the utility. The MMNL and regret-based models have also been used
in the context of route choice, e.g., Prato (2014) estimate PSL models using the regret model proposed by Chorus (2010), Bekhor et al. (2002) estimate an MMNL model based on the Error Component (EC) approach (Bolduc and Ben-Akiva, 1991) using route choice data collected in Boston, and Frejinger and Bierlaire (2007) also use the EC model with subnetwork components to capture the correlation between paths that share subnetworks in the transport network. We refer the reader to Frejinger (2008) and Prato (2009) for more details.

It is important to note that, thus far, we discussed only the first of the two aforementioned challenges associated with route choice modeling (i.e. definition of choice sets). Actually, the second challenge that consists of modeling correlated path utilities is closely related to the first one. While any model in Section 2.1 can be combined with consideration sets, this is not the case for models with sampled choice sets nor the recursive logit model. One of the main contributions of this thesis are recursive logit models that allows path utilities to be correlated.

2.4.1 Path-based Models with Sampled Choice Sets

The universal choice sets are huge or even infinite (if the network contains cycles), and cannot be enumerated. Sampling of alternatives then becomes an natural and appealing approach. The idea is to correct the choice probabilities so that the model can be consistently estimated based on sampled choice sets. The approach is easy and fast to apply with the MNL model, but when the randoms terms are correlated, it may however be difficult and time consuming in practice, both for estimating the model parameters and for making predictions. We provide in the following a detailed review of this approach focusing on models for uni-modal networks and reveled preference data.

Under sampling of alternatives, McFadden (1978) shows that the MNL can be consistently estimated using sampled choice sets, and Frejinger et al. (2009) successfully apply this approach to MNL-based route choice models. More precisely, for a sampled choice set \( D_n \), the corrected probability that a path \( i \) is chosen is (see for instance Section 2.1.5)

\[
P(i|D_n) = \frac{\exp \left( v_{ni} + \ln \frac{\kappa_{in}}{q(i)} \right)}{\sum_{j \in D_n} \exp \left( v_{nj} + \ln \frac{\kappa_{jn}}{q(j)} \right)},
\]

where \( q(i) \) is the sampling probability of path \( i \) and \( \kappa_{in} \) is the number of times alternative \( i \) is drawn. In this context, we note that Mai et al. (2015d) show that when the models are correctly specified, the information matrix equality holds if and only if the sampling corrections \( \ln \frac{\kappa_{in}}{q(i)} \), \( j \in D_n \), are added to the choice probabilities.

In order to deal with the overlapping of paths in the network, Frejinger et al. (2009) propose a heuristic sampling correction of the PS attribute, called expanded path size (EPS). This
attribute can be computed based on the lengths of links lying in the corresponding path and the expanded factors. Frejinger et al. (2009) estimated MNL route choice models based on a small and cycle-free network, but note that their methods have been used to quickly estimate a MNL model on real data using a real network of more than 7000 links (Mai et al., 2015c).

Other advanced MEV models have also been investigated for the route choice problem. Recently, Lai and Bierlaire (2015) have used the sampling of alternatives method proposed by Guevara and Ben-Akiva (2013a) for estimating a cross-nested route choice model, i.e., the Link-nested Logit (LNL) model proposed by Vovsha and Bekhor (1998). The idea of Lai and Bierlaire (2015) is to use the Metropolis-Hasting (MH) sampling algorithm (Flotterod and Bierlaire, 2013) to generate paths from an arbitrary distribution. The advantage of the MH algorithm is that it allows to sample a path \( i \) from a sampling probability \( q(i) = b(i) / \sum_{j \in C_n} b(j) = b(i) / B \), proportional to a weight \( b(i) \), without involving the term \( B \) which requires a sum over the full choice set \( C_n \). According to the sampling approach proposed by Guevara and Ben-Akiva (2013a), Lai and Bierlaire (2015) suggest a corrected choice probability for the cross-nested route choice model as

\[
P(i|D_n, D'_n, w) = \frac{\exp\left(v_{ni} + \ln G_i(D'_n, w) + \ln \frac{k_i}{b(i)}\right)}{\sum_{j \in D_n} \exp\left(v_{nj} + \ln G_j(D'_n, w) + \ln \frac{k_j}{b(j)}\right)},
\]

where \( D_n, D'_n \) are two samples of alternatives (\( D'_n \) is for the approximation of \( G_i \)), \( G \) is the CPGF of the cross-nested logit model, and \( G_i \) is its partial derivative with respect to \( \exp(v_{ni}) \), \( w \) is a vector of expansion factors to compensate the loss of the non-sampled paths of the nests. Lai and Bierlaire (2015) show that their models and methods are valid and practical by proving numerical results using synthetic and real data sets.

Finally, we note that the MMNL and regret-based models can be consistently estimated, thanks to the work of Guevara and Ben-Akiva (2013b) and Guevara et al. (2014). However, we are not aware of any MMNL or regret-based route choice model that is estimated using these sampling approaches.

### 2.4.2 Link-based Recursive Logit Model

The link-based recursive logit (RL) model (Fosgerau et al., 2013a) is a route choice model using the universal choice sets. The model is based on Rust’s dynamic discrete choice framework (Rust, 1987) where a path choice is modeled as a sequence of link choices. We describe the model in the following and refer the reader to the original paper (Fosgerau et al., 2013a) for more details.
We first introduce some notations. A transport network \( G(V; A) \) is considered, where \( V \) if the set of nodes (vertices) and \( A \) is the set of links (arcs). We denote the set of outgoing links from the sink node of a link \( k \in A \) by \( A(k) \). A path \( \sigma \) in this network is a sequence of links \((k_0; \ldots; k_J)\) with \( k_{j+1} \in A(k_j) \), for all \( 0 \leq j < J \).

In the RL model, each state is defined by a physical link. At each link/state, an action (or decision) consists of choosing a next link from a set of outgoing links. The route choice problem is then formulated as an infinite horizon dynamic programming problem with no discount. The model is developed using a simplified version of the Rust’s framework where the Markov transition probability matrix is degenerate (see for instance Section 2.1.6), i.e. given an action, the next state corresponds to the chosen link. So the RL model becomes a dynamic discrete choice model with a degenerate transition probability matrix. We note that, in this context, the route choice becomes a shortest path problem encoded into a parametric Markov decision process.

We associate with each outgoing link \( a \in A(k) \) of a given state/link \( k \) an instantaneous utility \( u(a|k) = v(a|k) + \mu \epsilon(a) \), where \( \mu \) is the scale of the random terms. The deterministic utility \( v(a|k) \) is assumed to be negative for all links except for a dummy link \( d \) that is zero. Dummy \( d \) is added to the network to define the destination as an absorbing state. At each choice stage the traveler maximizes the sum of the instantaneous and the expected maximum utility (value functions). The random terms are i.i.d. extreme value type I so that the choice model at each decision stage is the MNL. In this setting, the expected maximum utility from state \( k \) to the destination is recursively defined by Bellman’s equation

\[
V(k) = \mathbb{E}\left[ \max_{a \in A(k)} \left( v(a|k) + V(a) + \mu \epsilon(a) \right) \right] \quad \forall k \in A,
\]

The choice at each state is MNL, so the value function is recursively given by the logsum

\[
V(k) = \mu \ln \left( \sum_{a \in A(k)} e^{\frac{1}{\mu} (v(a|k) + V(a))} \right) \quad \forall k \in A, \tag{2.17}
\]

and the probability of choosing a next link \( a \in A(k) \) is simply given by the logit model

\[
P(a|k) = \frac{\exp \left( \frac{1}{\mu} (v(a|k) + V(a)) \right)}{\sum_{a \in A(k)} \exp \left( \frac{1}{\mu} (v(a|k) + V(a)) \right)} = \exp \left( \frac{1}{\mu} (v(a|k) + V(a) - V(k)) \right).
\]
A path $\sigma$ is a sequence links $(k_0, \ldots, k_J)$ with $k_{j+1} \in A(k_j)$ for all $0 \leq j < J$. The choice probability of path $\sigma$ can be computed as a sequence of link choices as

$$
P(\sigma, \beta) = \prod_{j=0}^{J-1} P(k_{j+1}|k_j) = \frac{\exp \left( \frac{1}{\mu} v(\sigma) \right)}{\exp \left( \frac{1}{\mu} V(k_0) \right)}, \quad (2.18)
$$

where $v(\sigma) = \sum_{j=0}^{J-1} v(k_{j+1}|k_j)$ is the deterministic utility of path $\sigma$. If we denote by $\Omega(k_0)$ the set of all feasible paths going from origin node $k_0$ to the destination (dummy link $d$), then the following equation is deduced from $(2.17)$

$$
\exp \left( \frac{1}{\mu} V(k_0) \right) = \sum_{\sigma \in \Omega(k_0)} \exp \left( \frac{1}{\mu} v(\sigma) \right). \quad (2.19)
$$

$(2.18)$ and $(2.19)$ suggest that the RL model is equivalent to the MNL model based on the choice set of all feasible paths in the network, as mentioned in Fosgerau et al. (2013a).

Fosgerau et al. (2013a) also propose a Link Size (LS) attribute in order to deterministically correct utilities for correlation. This attribute achieves an effect similar to the Path Size attribute in path-based models (Ben-Akiva and Bierlaire, 1999a). The models however retain the IIA property, unless the Path Size/Link Size attributes are updated as utilities change. Several studies in the literature (e.g. Bekhor et al., 2001, Frejinger and Bierlaire, 2007) report a better model fit and prediction results if the PS attributes are included in the deterministic utilities in addition to correlated random terms. It is also the case in the RL models as Fosgerau et al. (2013a) observe a significant improvement in final log-likelihood values when the LS attribute is used.

The RL model is convenient for the estimation in the sense that it does not require sampled choice sets, and the log-likelihood function can be computed quickly by solving systems of linear equations. This convenience is due to the CLOGIT assumption from the Rust’s model. However, under this assumption, the model is equivalent to the MNL model, hence exhibits the IIA property over paths in the network.

Finally, we note that the use of sequential link choice models to describe path choice has been investigated in the context of traffic assignment. For example, Dial (1971) present a traffic assignment method for uncongested networks where the demand on each origin-destination is distributed among routes according to an MNL model. This work however restricts the choices to a small set of “efficient” paths. Baillon and Cominetti (2008) propose a more general approach which includes the RL model, called Markovian traffic equilibrium, which is based on discrete choice models embedded in a dynamic programming framework. Baillon and Cominetti (2008) also show the existence of a unique equilibrium for their model and provide some numerical experiments. Melo (2012) presents a sequential choice model where a path is interpreted as a
bundle of goods, resulting in a Markovian assignment which is related to Baillon and Cominetti (2008).

2.4.3 Illustrative Example

In this section we use a small example to illustrate how different choice set formulations affect parameter estimates. More precisely, we focus on parameter estimates of the MNL model based on two assumptions on the choice sets: consideration sets and the universal choice sets. We consider a small network consisting of 15 nodes and two origins and one destination (see Figure 2.1). The network is cycle-free so that the universal choice sets can be enumerated. The numbers above or on the left of the links indicate the link travel times. In terms of travel time, we suppose that there are 13 attractive paths (5 paths from $O_1$ to $D$ and 8 from $O_2$ to $D$), which are in red while dashed paths are less attractive. The universal choice sets contains 15 paths from $O_1$ to $D$ and 21 paths going from $O_2$ to $D$, which are denoted by $U_1$ and $U_2$, respectively.

We specify the deterministic utility of each link based on travel time (TT) and a constant (LC). The latter equals one for each link, and the rationale behind using LC in the utilities is to penalize paths with many crossings. In Figure 2.2, we plot a histogram illustrating path utilities over the universal choice sets based on parameters $(\tilde{\beta}_{TT}, \tilde{\beta}_{LC}) = (-2.0; -0.1)$. The red bars refer to the attractive paths, i.e. 5 paths from $O_1$ to $D$ and 8 from $O_2$ to $D$, as mentioned above.
We consider four groups of choice sets. The first group, denoted by \((C_1^1, C_2^1)\), contains all the attractive paths, i.e., \(C_1^1\) contains the 5 attractive paths from \(O_1\) to \(D\) and \(C_2^1\) contains the 8 attractive paths from \(O_2\) to \(D\). The second and third groups \((C_1^2, C_2^2)\) and \((C_1^3, C_2^3)\) are created by adding to \((C_1^1, C_2^1)\) some less attractive paths according to the order of the path utilities (see in Figure 2.2). The last group refers to the universal choice sets \((U_1, U_2)\). We note that the numbers of alternatives in \(C_1^2\), \(C_2^2\), \(C_1^3\) and \(C_2^3\) are 8, 12, 12, 15, respectively.

In order to illustrate how the parameter estimates vary according to different choice set formulations we simulate two sets of observations. Each set consists of 500 observations for each OD pair based on parameters \((\hat{\beta}_{TT}, \hat{\beta}_{LC}) = (-2.0; -0.1)\). The first one is simulated based on the universal choice sets and the second is based on \((C_3^1, C_3^2)\). Tables 2.1 and 2.2 report the parameter estimates from the RL model, MNL-based PL model with sampling of alternatives \((PL^{sa})\) (Frejinger et al., 2009), and the PL model based on different consideration choice sets. We note that in the \(PL^{sa}\) model, paths are sampled and the choice probabilities are corrected using path probabilities given by the RL model, as suggested by Fosgerau et al. (2013a).

![Figure 2.2: Path utilities](image)

<table>
<thead>
<tr>
<th>(\hat{\beta}_{TT})</th>
<th>(\hat{\beta}_{LC})</th>
<th>(\beta_{TT})</th>
<th>(\beta_{LC})</th>
<th>(\beta_{TT})</th>
<th>(\beta_{LC})</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL</td>
<td>PL(^{sa}) (5 draws)</td>
<td>PL((C_1^1,C_2^1))</td>
<td>PL((C_1^2,C_2^2))</td>
<td>PL((C_1^3,C_2^3))</td>
<td>PL((U_1,U_2))</td>
</tr>
<tr>
<td>(-2.11) (0.11)</td>
<td>(-2.12) (0.12)</td>
<td>5.30 (0.30)</td>
<td>0.31 (0.22)</td>
<td>-2.02 (0.12)</td>
<td>-2.11 (0.11)</td>
</tr>
<tr>
<td>(-0.08) (0.05)</td>
<td>(-0.06) (0.06)</td>
<td>-0.02 (0.06)</td>
<td>0.47 (0.07)</td>
<td>-0.08 (0.05)</td>
<td>-0.08 (0.05)</td>
</tr>
</tbody>
</table>

Table 2.1: Numerical results when the true choice sets are \((U_1, U_2)\)

<table>
<thead>
<tr>
<th>(\hat{\beta}_{TT})</th>
<th>(\hat{\beta}_{LC})</th>
<th>(\beta_{TT})</th>
<th>(\beta_{LC})</th>
<th>(\beta_{TT})</th>
<th>(\beta_{LC})</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL</td>
<td>PL(^{sa}) (5 draws)</td>
<td>PL((C_1^1,C_2^1))</td>
<td>PL((C_1^2,C_2^2))</td>
<td>PL((C_1^3,C_2^3))</td>
<td>PL((U_1,U_2))</td>
</tr>
<tr>
<td>(-2.02) (0.09)</td>
<td>(-1.98) (0.10)</td>
<td>5.28 (0.24)</td>
<td>0.44 (0.18)</td>
<td>-1.90 (0.11)</td>
<td>-2.02 (0.09)</td>
</tr>
<tr>
<td>(-0.11) (0.05)</td>
<td>(-0.10) (0.05)</td>
<td>-0.02 (0.05)</td>
<td>0.43 (0.06)</td>
<td>-0.11 (0.05)</td>
<td>-0.11 (0.05)</td>
</tr>
</tbody>
</table>

Table 2.2: Numerical results when the true choice sets are \((C_1^3, C_2^3)\)

The results from Tables 2.1 and 2.2 show that the RL and PL\(^{sa}\) can retrieve the true parameters, even when the true choice sets is not the universal ones. The estimates given by the PL models depend on the choice sets. These models can only retrieve the true parameters when the choice sets are \((C_1^3, C_2^3)\), which are the true, or close to the true choice sets.
In summary, in this small example, we show that the parameter estimates given by the PL model with consideration choice sets may vary significantly depending on how choice sets are generated. This issue is also highlighted, e.g., in Frejinger et al. (2009). For the RL model and the sampling of alternatives approach, the estimates are consistent, even when the true choice sets are not the universal ones. In this case, the “unattractive” paths are assigned small probabilities and these probabilities do not significantly affect the parameter estimates.
Chapter 3

Nested Recursive Logit Model

In this chapter we propose a dynamic route choice model, called the nested recursive logit, NRL, that allows to relax the IIA property of the RL model. More precisely, we allow the scales of the random terms to be different over links, and the correlation between paths is given by the structure of the network. We propose a value iteration method with dynamic accuracy to quickly solve the resulting dynamic programming problem on a real network, and we estimate and analyze estimation and prediction results based on a real dataset collected in the Swedish city of Borlänge.

Some of the ideas from this chapter are also used in Chapter 8 to formulate MEV models as dynamic discrete choice models on networks of correlation structures. This formulation leads to a new approach to quickly estimate MEV models with large choice sets. The NRL model is also generalized in Chapter 4 where correlated random terms are taken into account.

The paper was presented at the 3rd Symposium of the European Association for Research in Transportation (hEART) (Leeds, UK) in September 2014, and was received the prize for best paper written by a Ph.D. student as lead author. This paper has been published as:

A Nested Recursive Logit Model for Route Choice Analysis

Tien Mai¹ Mogens Fosgerau² Emma Frejinger¹

Abstract

We propose a route choice model that relaxes the independence from irrelevant alternatives property of the logit model by allowing scale parameters to be link specific. Similar to the the recursive logit (RL) model proposed by Fosgerau et al. (2013a), the choice of path is modeled as a sequence of link choices and the model does not require any sampling of choice sets. Furthermore, the model can be consistently estimated and efficiently used for prediction.

A key challenge lies in the computation of the value functions, i.e., the expected maximum utility from any link in the network to a destination. The value functions are the solution to a system of non-linear equations. We propose an iterative method with dynamic accuracy that allows to efficiently solve these systems.

We report estimation results and a cross-validation study for a real network. The results show that the NRL model yields sensible parameter estimates and the fit is significantly better than the RL model. Moreover, the NRL model outperforms the RL model in terms of prediction.

Keywords: route choice modeling; nested recursive logit; substitution patterns; value iterations; maximum likelihood estimation; cross-validation.

3.1 Introduction

Discrete choice models are generally used for analyzing path choices in real networks based on revealed preference (RP) data. There are two main modeling issues associated with (i) estimating such models consistently and (ii) subsequently using them for prediction. First, choice sets of paths are unknown to the analyst and the set of all feasible paths for a given origin-destination pair cannot be enumerated. Second, path utilities may be correlated, for instance, due to physical overlap in the network. As we explain below, there is currently no path choice model that can be consistently estimated and used for prediction, while avoiding the specification of choice sets and allowing for correlation due to path overlap. The nested recursive logit (NRL) model, proposed in this paper, fills this gap.

¹Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
²Technical University of Denmark, Denmark, and Royal Institute of Technology, Sweden.
Most of the existing path choice models are based on choice sets of paths that need to be sampled before estimating or applying the model. Many different algorithms exist for sampling choice sets (for reviews, see e.g. Frejinger et al., 2009, Prato, 2009) and they all correspond to importance sampling protocols where paths have non-equal probabilities of being sampled. Frejinger et al. (2009) show that utilities need to be corrected for the sampling of alternatives, which implies that only algorithms that allow computation of the path sampling probabilities can be used. Frejinger et al. (2009) use the logit (MNL) model but recently Guevara and Ben-Akiva (2013a) and Guevara and Ben-Akiva (2013b) have derived results for generalized extreme value (GEV) and mixed logit models, respectively. The sampling approach can be used to consistently estimate a path choice model, but it is still unknown how to use that model for prediction.

A three path example network is often used to illustrate why it is important to allow for correlated utilities (we present this example in more detail in Section 3.3). At the origin one can take right or left. Going right there are two paths that share one link except for a short distance close to destination where they separate. If all three paths have the same deterministic utility, a logit model assigns the probability $1/3$ to each although one would expect a probability $1/2$ going left and $1/2$ going right. A number of models in the literature allow to model the correlation structure of path utilities. Examples are the link-nested logit (Vovsha and Bekhor, 1998), mixed logit with error components (Bekhor et al., 2001, Frejinger and Bierlaire, 2007) and paired combinatorial logit (Chu, 1989). These models are based on sampled choice sets without correcting the utilities for the sampling protocol. Hence, the parameter estimates are conditional on the choice sets and may have significantly different values if some paths are added or removed from the choice sets. This is problematic since the true choice sets are unknown. As mentioned earlier, the MEV models (e.g. link-nested logit) or the mixed logit models can be corrected. Lai and Bierlaire (2015) estimate a link-nested logit model using the results by Guevara and Ben-Akiva (2013a).

Recently, Fosgerau et al. (2013a) proposed the recursive logit (RL) model where path choice is modeled as a sequence of link choices using a dynamic discrete choice framework. The RL model can be consistently estimated and used for prediction without sampling choice sets of paths. It is however equivalent to a MNL model over the set of all feasible paths. A correction attribute called link size was proposed that achieves an effect similar to the path size attribute in path choice models (Ben-Akiva and Bierlaire, 1999a). These attributes correct the utilities for correlation but the models retain the independence of irrelevant alternatives (IIA) property, unless the path size/link size attributes are updated as utilities change (e.g. changes in link travel times).

In this paper we propose an extension of the RL model that allows path utilities to be correlated in a fashion similar to nested logit (Ben-Akiva, 1973, McFadden, 1978) and where links can
have different scale parameters. The key challenge with this extension lies in the computation of the expected maximum utility from a current position in the network until the destination (value functions). A computational advantage of the RL model is that the value functions can be computed by solving a system of linear equations, which is fast and easy to do. In the case of the NRL, the value functions are a solution to a system of non-linear equations which is substantially more difficult to deal with. We propose an iterative method with dynamic accuracy to efficiently solve this equation system.

This paper makes a number of contributions. First we propose a model that can be consistently estimated and used for prediction without sampling choice sets while allowing the random terms to be correlated. Second, we provide illustrative examples and discuss substitution patterns in order to build an intuition on the properties of the model. Third, we propose an iterative method to solve for the value functions and we derive the analytical gradient of the log-likelihood function for the case that the scales are functions of model parameters so that the NRL model can be efficiently estimated. Fourth, we provide estimation and cross-validation results for a real network using simulated and real observations. Finally, the estimation code is implemented in MATLAB and is freely available upon request.

The paper is structured as follows. Section 3.2 presents the NRL model. Section 3.3 discusses substitution patterns by illustrative examples and Section 3.4 provide a method to compute the value functions. Section 3.5 derives an analytical formula for the first order derivative of the log-likelihood function. Specifications, estimation and prediction results are presented in Section 3.6 and finally Section 3.7 concludes.

### 3.2 The Nested Recursive Logit Model

In the RL model (Fosgerau et al., 2013a) the path choice problem is formulated as a sequence of link choices and modeled in a dynamic discrete choice framework. At each node the decision maker chooses the utility-maximizing outgoing link with link utilities given by the instantaneous utility and the expected maximum utility to the destination. The random terms of the instantaneous utilities are independently and identically distributed (i.i.d.) extreme value type I so that the model is equivalent to MNL. In this section we present the NRL model which relaxes the IIA property of MNL by assuming that the scales of random terms are non-equal across links. We derive the NRL model using the same notation as Fosgerau et al. (2013a) (we refer the reader to that paper for a more detailed presentation of the notation). Even though the derivation of NRL is similar to the RL one, the resulting expressions of the value functions and path choice probabilities have important differences.
A directed connected graph (not assumed acyclic) $G = (A, V)$ is considered, where $A$ and $V$ are the sets of links and nodes, respectively. For each link $k \in A$, we denote the set of outgoing links from the sink node of $k$ by $A(k)$. Moreover we associate an absorbing state with each destination by extending the network with dummy links $d$ (see Figure 3.1). This is a link without successors so a trip stops once this state is reached. The set of all links is $\tilde{A} = A \cup \{d\}$ and the corresponding deterministic utility is $v(d|k) = 0$ for all $k$ that have destination $d$ as sink node. Given two links $a, k \in \tilde{A}$, the following instantaneous utility is associated with action $a \in A(k)$ of individual $n$

$$u^n(a|k; \beta) = v^n(a|k; \beta) + \mu_k \epsilon(a)$$

(3.1)

where $\beta$ is a vector of parameters, $\epsilon(a)$ are i.i.d extreme value type I and $\mu_k$ is a strictly positive scale parameter. We ensure that $\epsilon(a)$ have zero mean by subtracting Euler’s constant. The deterministic term $v^n(a|k; \beta)$ is assumed negative for all links except the dummy link $d$. We emphasize the difference with the original RL model where scale parameters are assumed equal ($\mu_k = \mu \forall k \in A$). For notational simplicity, we omit an index for individual $n$ but note that the utilities can be individual specific.

The expected maximum utility from the sink node of $k$ to the destination is the value function $V^d(k; \beta)$. The superscript $d$ indicates that the value functions are destination specific and they also depend on parameters $\beta$. $V^d(k; \beta)$ is recursively defined by Bellman’s equation

$$V^d(k; \beta) = E \left[ \max_{a \in A(k)} \left( v(a|k; \beta) + V^d(a; \beta) + \mu_k \epsilon(a) \right) \right] \quad \forall k \in A$$

(3.2)

or equivalently

$$\frac{1}{\mu_k} V^d(k; \beta) = E \left[ \max_{a \in A(k)} \left( \frac{1}{\mu_k} v(a|k; \beta) + V^d(a; \beta) + \epsilon(a) \right) \right] \quad \forall k \in A. \quad (3.3)$$

For notational simplicity we omit from now on $\beta$ from the value functions $V(\cdot)$ and the utilities $v(\cdot)$.
Given these assumptions the probability of choosing link $a$ given state $k$ is given by the MNL model

$$P_d(a|k) = \delta(a|k) \frac{\frac{1}{\mu_k} (v(a|k) + V^d(a))}{\sum_{a' \in A(k)} e^{\frac{1}{\mu_k} (v(a'|k) + V^d(a'))}}$$

$$= \delta(a|k) e^{\frac{1}{\mu_k} (v(a|k) + V^d(a)) - V^d(k)} \quad \forall k, a \in \tilde{A}.$$  

(3.4)

Note that we include $\delta(a|k)$ that equals one if $a \in A(k)$ and zero otherwise so that the probability is defined for all $a, k \in \tilde{A}$ (we recall that $\tilde{A} = A \cup \{d\}$). Since we assume that the random terms in (3.1) are distributed i.i.d. EV type I, the value functions (3.2) are given recursively by the logsum

$$\frac{1}{\mu_k} V^d(k) = \ln \left( \sum_{a \in A(k)} e^{\frac{1}{\mu_k} (v(a|k) + V^d(a))} \right) \quad \forall k \in A$$

(3.5)

and $V^d(d) = 0$ by assumption. Similar to Fosgerau et al. (2013a) we can write (3.5) as

$$\frac{1}{\mu_k} V^d(k) = \begin{cases} 
\sum_{a \in A} \delta(a|k) e^{\frac{1}{\mu_k} (v(a|k) + V^d(a))} & \forall k \in A \\
1 & k = d
\end{cases}$$

(3.6)

and define a matrix $M^d(|\tilde{A}| \times |\tilde{A}|)$ and a vector $z^d(|\tilde{A}| \times 1)$ with entries

$$M^d_{ka} = \delta(a|k) e^{\frac{V_d(a)}{\mu_k}}, \quad z^d_{k} = e^{\frac{V_d(k)}{\mu_k}}, \quad k, a \in \tilde{A}.$$  

(3.7)

The key issue here compared to the RL model is that we do not end up with a system of linear equations. Indeed, the value functions are the solutions to the following system of non-linear equations

$$z^d_k = \begin{cases} 
\sum_{a \in A} M^d_{ka} (z^d_{a})^{\mu_a/\mu_k} & \forall k \in A \\
1 & k = d
\end{cases}$$

(3.8)

where the non-linearity arises due to the scale parameters $\mu_k$ not being equal.

The probability of a path $\sigma$ defined by a sequence of links $\sigma = [k_0, k_1, \ldots, k_I]$ is

$$P(\sigma) = \prod_{i=0}^{I-1} e^{\frac{1}{\mu_{k_i}} (v(k_{i+1}|k_i) + V^d(k_{i+1}) - V^d(k_i))}. \quad (3.9)$$

Unlike the RL model, the link specific value functions do not cancel out due to the scale parameters. This implies that the path choice probabilities are computationally more costly to evaluate.

We note that if the network contains cycles, the RL and NRL model allow for loops (Akamatsu, 1996, Fosgerau et al., 2013a, discuss this in more detail). The probability of paths with loops
depend on the data and network structure. For the data used in this paper, Fosgerau et al. (2013a) report that paths with loops have a very small probability.

Finally we note that the IIA property does not hold in the NRL model. Consider the ratio of the choice probabilities of two paths \( \sigma_1 = [k_1, \ldots, k_{I_1}] \) and \( \sigma_2 = [h_1, \ldots, h_{I_2}] \) connecting just one origin-destination pair

\[
\frac{P(\sigma_1)}{P(\sigma_2)} = \frac{\prod_{i=1}^{I_1-1} e^{\mu_{k_i}} (v(k_{i+1} | k_i) + V^d(k_{i+1}) - V^d(k_i))}{\prod_{i=1}^{I_2-1} e^{\mu_{h_i}} (v(h_{i+1} | h_i) + V^d(h_{i+1}) - V^d(h_i))}.
\]

When the scales \( \mu_k = \mu \ \forall k \in A \), the value function terms cancel out and the ratio (3.10) then only depends on the utilities of two considered paths. For the NRL model, the ratio (3.10) depends on several values functions, which are evaluated based on the whole network and therefore the IIA property does not hold. In the following section we discuss the resulting substitution pattern in more depth using several illustrative examples.

### 3.3 Illustrative Examples and Substitution Patterns

Similar to several studies in the literature (e.g. Ben-Akiva and Bierlaire, 1999a), we use a simple three path network shown in 3.2 to illustrate why it is important to allow for correlated utilities. There are three paths from \( o \) to \( d \) (link \( o \) is the origin and link \( d \) is the destination dummy link): \([o, a, d] \), \([o, b, e, d] \), \([o, b, f, d] \). We number these paths 1, 2 and 3 and the corresponding path probabilities are \( P_1 \), \( P_2 \) and \( P_3 \), respectively. The only attribute in the instantaneous utility is link length and the values are given in the parentheses on each arc. In order to compute path probabilities we choose a length parameter \( \tilde{\beta} = -1 \).

When the scales \( \mu_k = \mu \ \forall k \in A \), the model corresponds the RL and \( P_1 = P_2 = P_3 = 1/3 \). When the network has a perfect nested structure as this one (each path in the network belongs to exactly one nest when defined by physical overlap), the NRL model is equivalent to a nested logit model. We can illustrate this by fixing all the scale parameters to 1 except \( \mu_b \) that we vary over the interval \((0, 1)\). The path probabilities are plotted in the graph on the right hand side in Figure 3.2.

In order to build intuition on the substitution patterns implied by the NRL model, we provide three more examples. The first is shown in Figure 3.3 which also has a simple nested structure. There are 4 nodes \( A, B, C, D \) and 9 links. Moreover, there are 6 possible paths from \( o \) to \( d \): \([o, a, a_1, d] \), \([o, a, a_2, d] \), \([o, a, a_3, d] \), \([o, b, b_1, d] \), \([o, b, b_2, d] \) and \([o, b, b_3, d] \) and we number these paths as 1, 2, 3, 4, 5 and 6, respectively.
Figure 3.2: Classic three paths example network

For the RL model the IIA property holds, meaning that, if we remove any link in the network, the probabilities of the remaining feasible paths will increase by the same proportion (for example if we remove link $a_2$, the probabilities of path $[o, a, a_3, d]$ and path $[o, b, b_3, d]$ increase but they are still equal). For the NRL model, the scales of random terms are assigned different values. We assign a scale of 0.5 for links $a$, a scale of 0.8 for links $b$ and a scale of 1.0 for the others. Similar to an example in Train (2003), we illustrate substitution patterns by removing in turn links $a_1, a_2, b_1, b_2$ and present changes in probabilities in Table 3.1.

Figure 3.3: Example network with perfect nested structure

<table>
<thead>
<tr>
<th>Paths</th>
<th>Original</th>
<th>Probabilities with link removed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td>1 : $[o, a, a_1, d]$</td>
<td>0.54</td>
<td>-</td>
</tr>
<tr>
<td>2 : $[o, a, a_2, d]$</td>
<td>0.15</td>
<td>0.38(+151%)</td>
</tr>
<tr>
<td>3 : $[o, a, a_3, d]$</td>
<td>0.04</td>
<td>0.11(+151%)</td>
</tr>
<tr>
<td>4 : $[o, b, b_1, d]$</td>
<td>0.02</td>
<td>0.05(+93%)</td>
</tr>
<tr>
<td>5 : $[o, b, b_2, d]$</td>
<td>0.06</td>
<td>0.12(+93%)</td>
</tr>
<tr>
<td>6 : $[o, b, b_3, d]$</td>
<td>0.17</td>
<td>0.33(+93%)</td>
</tr>
</tbody>
</table>

Table 3.1: Change in probability when link is removed (example network with perfect nested structure)

We note that the probabilities for paths $[o, a, a_1, d]$, $[o, a, a_2, d]$, $[o, a, a_3, d]$ rise by the same proportions whenever one link is removed from the network. This is also the case for the three paths $[o, b, b_1, d]$, $[o, b, b_2, d]$ and $[o, b, b_2, d]$. As expected, the IIA property holds between paths within the same nest but not for paths in different nests. For example, when link $a_1$ is removed,
the probabilities of the paths in the first nest rise by 151% while the paths in the second nest rise by 93%.

We also consider the case when a link from node B to C is added to the network in Figure 3.3. This change adds three more paths to nest N1. In Table 3.2 we report the change in probabilities for the same six paths as before. We note that the absolute values of choice probabilities change but the substitution pattern remains proportional.

Table 3.2: Change in probability when link is removed (example network with perfect nested structure with link from B to C)

<table>
<thead>
<tr>
<th>Paths</th>
<th>Original</th>
<th>Probabilities with link removed</th>
<th>a1</th>
<th>a2</th>
<th>b1</th>
<th>b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : [o, a, a1, d]</td>
<td>0.487</td>
<td>0.572(17.52%)</td>
<td>0.504(3.48%)</td>
<td>0.522(7.27%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 : [o, a, a2, d]</td>
<td>0.140</td>
<td>0.298(113.38%)</td>
<td>-</td>
<td>0.144(3.48%)</td>
<td>0.150(7.27%)</td>
<td></td>
</tr>
<tr>
<td>3 : [o, a, a3, d]</td>
<td>0.040</td>
<td>0.085(113.38%)</td>
<td>0.047(17.52%)</td>
<td>0.041(3.48%)</td>
<td>0.043(7.27%)</td>
<td></td>
</tr>
<tr>
<td>4 : [o, b, b1, d]</td>
<td>0.022</td>
<td>0.038(73.88%)</td>
<td>0.024(12.84%)</td>
<td>-</td>
<td>0.026(22.29%)</td>
<td></td>
</tr>
<tr>
<td>5 : [o, b, b2, d]</td>
<td>0.059</td>
<td>0.102(73.88%)</td>
<td>0.066(12.84%)</td>
<td>0.063(7.86%)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>6 : [o, b, b3, d]</td>
<td>0.160</td>
<td>0.278(73.88%)</td>
<td>0.180(12.84%)</td>
<td>0.172(7.86%)</td>
<td>0.195(22.29%)</td>
<td></td>
</tr>
</tbody>
</table>

The network in Figure 3.3 is designed so that the paths can naturally be divided into separate nests. In the next example shown in Figure 3.4 we slightly modify the network so that paths have a cross-nested structure. More precisely, we add a node E that splits links a3 and b1 into two links. The lengths of the paths in the new network do not change but the structure of the network is different since apart from the origin and destination, two paths [o, a, a3, e, d] and [o, a, b1, e, d] share link e. Furthermore, there is a new link f going (backward) from node E to node A so that the expected maximum utilities from link a3 and b1 depend on the whole network.

We report probabilities for the 6 paths without loops: [o, a, a1, d], [o, a, a2, d], [o, a, a3, e, d], [o, b, b1, e, d], [o, b, b2, d], [o, b, b3, d], which are numbered as 1, 2, 3, 4, 5 and 6, respectively. We keep the same scales as in the first example (i.e. \( \mu_a = 0.5 \), \( \mu_b = 0.8 \) and the other scale parameters are equal to one). The changes in probabilities of the six paths when we remove in turn links a3, b1 and f are reported in Table 3.3. We note that the substitution patterns are different than in the previous example since the probabilities of paths 3 and 4 no longer change by the same proportion as the other paths in their respective nest.
In order to compare the results with path based models we report probabilities given by the nested logit and link-nested logit (Vovsha and Bekhor, 1998) models in Table 3.4. The correlation structure given by the link-nested logit model is shown in Figure 3.5. For the nested models, the nesting parameters take the same values as in the NRL mode, namely 0.8 for nest \( N_1 \) and 0.5 for nest \( N_2 \). The results show that for these examples, the probabilities of the nested model are identical to the NRL model and probabilities of the link-nested logit are slightly different from NRL. We note that the sums of the path probabilities for RL and NRL in the second example are slightly smaller than one, due to the cycle in the network.

In summary, the IIA property can be relaxed by assuming different scales. The resulting substitution pattern depends on the network structure. If the network has a perfect nested structure (e.g Figure 3.3) the NRL and nested logit models yield the same results.

<table>
<thead>
<tr>
<th>Paths</th>
<th>Original</th>
<th>( a_1 )</th>
<th>( b_3 )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : ([o, a, a_1, d])</td>
<td>0.54</td>
<td>-</td>
<td>0.60(+12%)</td>
<td>0.54(+0.7%)</td>
</tr>
<tr>
<td>2 : ([o, a, a_2, d])</td>
<td>0.15</td>
<td>0.38(+150%)</td>
<td>0.17(+12%)</td>
<td>0.15(+0.7%)</td>
</tr>
<tr>
<td>3 : ([o, a, a_3, e, d])</td>
<td>0.05</td>
<td>0.11(+148%)</td>
<td>0.05(+11%)</td>
<td>0.04(-1.3%)</td>
</tr>
<tr>
<td>4 : ([o, b, b_1, e, d])</td>
<td>0.03</td>
<td>0.05(+86%)</td>
<td>0.05(+90%)</td>
<td>0.02(-6.7%)</td>
</tr>
<tr>
<td>5 : ([o, b, b_2, d])</td>
<td>0.06</td>
<td>0.12(+93%)</td>
<td>0.12(+91%)</td>
<td>0.06(+1.4%)</td>
</tr>
<tr>
<td>6 : ([o, b, b_3, d])</td>
<td>0.17</td>
<td>0.33(+93%)</td>
<td>-</td>
<td>0.17(+1.4%)</td>
</tr>
</tbody>
</table>

Table 3.3: Change in probability when link is removed (example network with cross-nested structure)

![Cross-nested structure from the Link-nested logit model](image)

Figure 3.5: Cross-nested structure from the Link-nested logit model

<table>
<thead>
<tr>
<th>Paths</th>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNL</td>
<td>NRL</td>
</tr>
<tr>
<td>1</td>
<td>0.449</td>
<td>0.541</td>
</tr>
<tr>
<td>2</td>
<td>0.165</td>
<td>0.155</td>
</tr>
<tr>
<td>3</td>
<td>0.061</td>
<td>0.044</td>
</tr>
<tr>
<td>4</td>
<td>0.061</td>
<td>0.023</td>
</tr>
<tr>
<td>5</td>
<td>0.100</td>
<td>0.064</td>
</tr>
<tr>
<td>6</td>
<td>0.165</td>
<td>0.173</td>
</tr>
</tbody>
</table>

Table 3.4: Path probabilities comparison
3.4 Computation of the Value Functions

The main challenge associated with the NRL model is to efficiently solve the large-scale system of non-linear equations (3.6). In the following we describe a value iteration approach that is efficient thanks to (i) a good initial solution and (ii) dynamic accuracy.

We define a matrix $X(z)$ with entries

$$X(z)_{ka} = z_a^{\mu_k/\mu_k} \quad \forall k, a \in \tilde{A}$$

so that the Bellman equation (3.8) can be written as

$$z = [M \circ X(z)]e + b.$$  \hspace{1cm} (3.12)

$b$ is a vector of size $(|\tilde{A}| \times 1)$ with zero values for all states except for the destination that equals $1$, $e$ is a vector of size $(|\tilde{A}| \times 1)$ with value one for all states and $\circ$ is the element-by-element product.

Value iterations are based on Equation (3.12). We start with an initial vector $z^0$ and then for each iteration $i$ we compute a new vector

$$z^{i+1} \leftarrow [M \circ X(z^i)]e + b.$$  \hspace{1cm} (3.13)

and iterate until a fixed point is found using $||z^{i+1} - z^i||^2 < \gamma$ for a given threshold $\gamma > 0$ as stopping criteria.\footnote{It can be shown that if the Bellman equation has a solution, this method converges after a finite number of iterations (see for instance Rust, 1987, 1988). The choice of initial vector is however important for the rate of convergence. We use the solution of the system of linear equations corresponding to the RL model ($\mu_k = \mu \forall k \in A$) which is fast to compute.}

The proofs in the literature establishing the existence and uniqueness of a solution to Bellman’s equation use a discount factor less than one. In our case we do not discount future utilities and these proofs do not apply. Fosgerau et al. (2013a) discuss this issue in more detail for the RL model. In essence, the existence of a solution depends on the balance between the number of paths connecting the nodes in the network and the size of the scaled instantaneous utilities. It is easy to find a feasible solution by using large enough magnitude of the $\beta$ parameters.

Since the value functions depend on the parameter values, they need to be solved repeatedly when searching over the parameter space (maximum likelihood estimation). In order to decrease the computational time we use dynamic accuracy. More precisely, we update the threshold $\gamma$ in

$$\sum_{k \in \tilde{A}} (V^{i+1}(k) - V^i(k))^2 < \gamma'.$$ The value functions have however larger magnitudes than $z$.\footnote{The value functions can also be used in the stopping criteria i.e. the iteration stops when $\sum_{k \in \tilde{A}} (V^{i+1}(k) - V^i(k))^2 < \gamma'$. The value functions have however larger magnitudes than $z$.}
the iterations of the non-linear optimization algorithm so that higher accuracy is required close to optimum (\( \gamma \) decreases as the number of iterations of the non-linear optimization algorithm increases).

Before discussing the maximum likelihood estimation in more detail, we note that (3.12) can be written as \( F(z) = 0 \), where \( F(z) = z - [M \circ X(z)]e + b \). A standard solver can be used e.g. \texttt{fsolver} in MATLAB or the Newton-GMRES method (for instance Kelley, 1995). We have tested these methods but found that they are not efficient for our application and that our approach works better.

### 3.5 Maximum Likelihood Estimation

There are several different ways of estimating a dynamic discrete choice model (Aguirregabiria and Mira, 2010), we adopt the nested fixed point algorithm of Rust (1987). This algorithm combines an outer iterative non-linear optimization algorithm for searching over the parameter space with an inner algorithm for solving the value functions.\footnote{Another option is the algorithm proposed by Aguirregabiria and Mira (2002). The idea is to swap the order of the outer and inner algorithms so that the outer algorithm solves the value functions and the inner algorithm maximizes the pseudo-likelihood function. This is very useful if the value functions are costly to evaluate. In the case of the NRL model, it is more costly to maximize the log-likelihood function than solving the value functions.}

The latter was the focus of the previous section and we now turn our attention to the definition of the log-likelihood (LL) function and the derivation of its gradient which allows us to use classic Hessian approximation such as BHHH and BFGS (see for instance Berndt et al., 1974, Nocedal and Wright, 2006).

The path probabilities are defined by (3.9) and contain scale parameters \( \mu_k \) for all \( k \in A \) as well as the parameters \( \beta \) associated with the attributes of the instantaneous utilities. Clearly, it is not possible to estimate all link-specific scale parameters for a real network and therefore we assume that they are a function of parameters \( \beta \) to be estimated \( \mu_k(\beta) \). (We refer the reader to the numerical results, Section 3.6, for an example.)

The LL function defined over the set of path observations \( n = 1, \ldots, N \) is

\[
LL(\beta) = \sum_{n=1}^{N} \ln P(\sigma_n, \beta) = \sum_{n=1}^{N} \sum_{t=0}^{I_n} \frac{1}{\mu_{k_t}} (e^n(k_{t+1}|k_t) + V^n(k_{t+1}) - V^n(k_t))
\]

(3.14)

and is very similar to the LL function of the RL model except that the value functions for the states along a path do not cancel out. Assuming a linear-in-parameters formulation of the
instantaneous utilities, the gradient with respect to a given parameter $\beta_i$ is

$$\frac{\partial LL(\beta)}{\partial \beta_i} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n-1} \frac{1}{\mu_{k_t}} \left( \frac{\partial v^n(k_{t+1}|k_t)}{\partial \beta_i} + \frac{\partial V^n(k_{t+1})}{\partial \beta_i} - \frac{\partial V^n(k_t)}{\partial \beta_i} \right)$$

$$- \frac{\partial \mu_{k_t}}{\mu_{k_t}^2} \frac{1}{\partial \beta_i} \left( v^n(k_{t+1}|k_t) + V^n(k_{t+1}) + V^n(k_t) \right)$$

and hence requires the first derivative of the value functions $V^n(k)$, $\forall k \in \tilde{A}$ with respect to $\beta_i$.

We define $\phi_{ka} = \mu_a / \mu_k$ and take the derivative of a given value function $z_k$ as defined by (3.8) (without using the superscript for destination $d$) and obtain

$$\frac{\partial z_k}{\partial \beta_i} = \sum_{a \in \mathcal{A}} \left( \frac{\partial M_{ka} \phi_{ka}}{\partial \beta_i} z_a + M_{ka} \phi_{ka} \frac{\partial z_a}{\partial \beta_i} + \frac{\partial \phi_{ka}}{\partial \beta_i} \ln z_a \right)$$

$$= \sum_{a \in \mathcal{A}} \left( \frac{\partial M_{ka} \phi_{ka}}{\partial \beta_i} z_a + M_{ka} \phi_{ka} \frac{\partial \phi_{ka}}{\partial \beta_i} \ln z_a \right) + \sum_{a \in \mathcal{A}} \left( M_{ka} \phi_{ka} \frac{\partial \phi_{ka}}{\partial \beta_i} \ln z_a \right).$$

(3.15)

We note that when the scales $\mu_k$ contain some model parameters, the derivative of each element of matrix $M(\beta)$ with respect to a given parameter $\beta_i$ is

$$\frac{\partial M_{ka}}{\partial \beta_i} = \delta(a|k) e^{\frac{v(a|k)}{\mu_k}} \left( \frac{\partial v(a|k)}{\partial \beta_i} - \frac{v(a|k)}{\mu_k} \frac{\partial \mu_k}{\partial \beta_i} \right), \quad k, a \in \tilde{A}.$$

We introduce two matrices, $G^i$ and $K$ of size $|\tilde{A}| \times |\tilde{A}|$, which have the two sums of (3.15) as entries

$$G^i_{ka} = \frac{\partial M_{ka} \phi_{ka}}{\partial \beta_i} z_a + M_{ka} \phi_{ka} \frac{\partial \phi_{ka}}{\partial \beta_i} \ln z_a$$

$$K_{ka} = M_{ka} \phi_{ka} \frac{\partial \phi_{ka}}{\partial \beta_i} z_a, \quad \forall k, a \in \tilde{A}.$$

(3.16)

This allows us to define the Jacobian of vector $z$ as a system of linear equations

$$\frac{\partial z}{\partial \beta_i} = G^i e + K \frac{\partial z}{\partial \beta_i} \Rightarrow \frac{\partial z}{\partial \beta_i} = (I - K)^{-1} G^i e,$$

(3.17)

which in theory, can be solved very efficiently. Nevertheless, it is possible to use the fact that $V(k) = \mu_k \ln z_k \forall k \in \tilde{A}$ and derive the Jacobian of $V$ instead of $z$. In this case the gradient of $V(k)$ with respect to a given $\beta_i$ is

$$\frac{\partial V(k)}{\partial \beta_i} = \frac{\partial \mu_k}{\partial \beta_i} \ln z_k + \frac{\mu_k}{z_k} \frac{\partial z_k}{\partial \beta_i}.$$

(3.18)

Using (3.15) we get

$$\frac{\partial V(k)}{\partial \beta_i} = \sum_{a \in \mathcal{A}} S_{ka}^i + \sum_{a \in \mathcal{A}} H_{ka} \frac{\partial V(a)}{\partial \beta_i} + h_k$$

(3.19)
where
\[ S_{ka}^i = \mu_k \frac{\partial M_{ka}}{\partial \beta_i} \frac{z_a}{z_k} + \mu_k M_{ka} \ln(z_a) \frac{z_a}{z_k} \frac{\partial \phi_{ka}}{\partial \beta_i} - M_{ka} \ln(z_a) \frac{z_a}{z_k} \frac{\partial \mu_a}{\partial \beta_i} \]
and
\[ H_{ka} = M_{ka} \frac{z_a}{z_k} \phi_{ka} \]
and \( h_k = \frac{\partial \mu_k}{\partial \beta_i} \ln z_k \).

We denote \( S^i, H \) be two matrices of size \(|\hat{A}| \times |\hat{A}|\) and \( h, V \) be two vectors of size \(|\hat{A}| \times 1\) with entries \( S_{ka}^i, H_{ka}, h_k, V(k) \) for all \( k,a \in \hat{A} \), respectively. The Jacobian of vector \( V \) can then be written as a system of linear equations
\[
\frac{\partial V}{\partial \beta_i} = (I - H)^{-1} (S^i c + h). \tag{3.20}
\]

Although theoretically equivalent, we now discuss the numerical differences between the two formulas (3.17) and (3.20) for computing the gradient of the value functions. We consider the definitions of the matrix \( K \) and \( H \). \( z_a, a \in \hat{A} \) are exponential functions of the value functions which are negative by assumption. The value of \( z_a \) may therefore be very close to zero. Since the elements of matrix \( K \) can be written as \( K_{ka} = \phi_{ka} M_{ka} z_a^{-1} (\forall k,a \in \hat{A}) \) if \( \phi_{ka} < 1 \), the value of \( K_{ka} \) can be very large, and if \( \phi_{ka} > 1 \), \( K_{ka} \) can be very close to zero. These wide range of values in the elements of matrix \( K \) (and also in matrix \( I - K \)) can lead to numerical issues when solving the system (3.17). Based on equation (3.8), each element of matrix \( H \) can be written as
\[
H_{ka} = \frac{M_{ka} z_a^{-1}}{\sum_{a' \in A(k)} M_{ka'} z_{a'}^{-1}} = \frac{1}{1 + \sum_{a' \in A(k), a' \neq a} \frac{M_{ka} z_a}{M_{ka'} z_{a'}}} \quad \forall k,a \in \hat{A}, a \in A(k)
\]
so that \( 0 < H_{ka} < 1 \), meaning that the elements of matrix \( H \) are closer in value, compared to matrix \( K \). Therefore, using (3.20) to compute the gradient of LL function is better than (3.17) for numerical reasons. In summary, the analytical gradient of the LL function has a complicated form but can be efficiently computed by solving systems of linear equations.

### 3.6 Numerical Results

In this section we present estimation and prediction results for four different models: the RL model with and without link size (LS) attribute and the NRL model, also with and without LS attribute. We use the same data as Fosgerau et al. (2013a) (also used in Frejinger and Bierlaire, 2007, Mai et al., 2015d) which has been collected in Borlänge, Sweden. The network is composed of 3077 nodes and 7459 links and is uncongested so travel times can be assumed...
static and deterministic. The sample consists of 1832 trips corresponding to simple paths with a minimum of five links. Moreover, there are 466 destinations, 1420 different origin-destination (OD) pairs and more than 37,000 link choices in this sample.

### 3.6.1 Model Specifications

The same five attributes as Fosgerau et al. (2013a) are used in the instantaneous utilities. First, link travel time $TT(a)$ of action $a$. Second, a left turn dummy $LT(a|k)$ that equals one if the turn angle from $k$ to $a$ is larger than 40 degrees and less than 177 degrees. Third, a u-turn dummy $UT(a|k)$ that equals one if the turn angle is larger than 177. Fourth, a link constant $LC(a)$. The fifth attribute is $LS(a)$ (for a detailed description see Fosgerau et al., 2013a) and it has been computed using a linear-in-parameters formulation of the aforementioned four attributes using parameters $\tilde{\beta}_{TT} = -2.5$, $\tilde{\beta}_{LT} = -1$, $\tilde{\beta}_{LC} = 0.4$, $\tilde{\beta}_{UT} = -4$.

Even in this fairly small network there are more than 7000 links, so it is not possible to estimate link specific parameters. We therefore impose a constraint on the scale parameters $\mu_k > 0$ by defining them as a function of link attributes. More precisely, $\mu_k = e^{\lambda_k}$ where $\lambda_k = \omega x_k$, $\omega$ is a vector of parameters and $x_k$ a vector of attributes associated with link $k$. This assumption ensures that (i) the estimation problem is unconstrained and (ii) we can use the analytical gradient (3.18). Note that if all the parameters in $\lambda_k$ are zero, the scales are equal to one for all links $k \in \tilde{A}$, meaning that the NRL model becomes the RL model. As much as data allows, it is possible to elaborate on the specification of the scale parameters. For example, by including different attributes in the exponential function or by estimating link specific scales parameters for some links in the network.

For the numerical results presented in this paper we use three link specific attributes: travel time, LS and the number of outgoing links $OL(k) = |A(k)|$. Accordingly, $\lambda_k$ is

$$
\lambda_k = \omega_{TT} TT(k) + \omega_{LS} LS(k) + \omega_{OL} OL(k).
$$

We do not use a link constant since it has the same value for all links, the rationale behind using it in the instantaneous utilities is to penalize paths with many crossings (links). Note that this is not a regression model, it is simply a specification of the scale parameters $\mu_k$ that enter the instantaneous utility functions.
To summarize, the deterministic utilities for four different model specifications with respect to link \( a \) given link \( k \) are

\[
v_{RL}(a|k; \beta) = v_{NRL}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k)
\]

\[
v_{RL-LS}(a|k; \beta) = v_{NRL-LS}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k) + \beta_{LS}LS(a)
\]

and the instantaneous utilities are

\[
u_{RL}(a|k; \beta) = v_{RL}(a|k; \beta) + \mu \epsilon(a)
\]

\[
u_{RL-LS}(a|k; \beta) = v_{RL-LS}(a|k; \beta) + \mu \epsilon(a)
\]

\[
u_{NRL}(a|k; \beta, \omega) = v_{NRL}(a|k; \beta) + e^{\lambda_k} \epsilon(a)
\]

\[
u_{NRL-LS}(a|k; \beta, \omega) = v_{NRL-LS}(a|k; \beta) + e^{\lambda_k} \epsilon(a).
\]

### 3.6.2 Estimation Results

We report the estimation results for the four specifications in Table 3.5. The results are comparable to those previously published using the same data. The \( \beta \) estimates have their expected signs and are highly significant. \( \hat{\omega}_{LS} \) and \( \hat{\omega}_{OL} \) are significant and negative while \( \hat{\omega}_{TT} \) is not significantly different from zero when the LS attribute is included in the instantaneous utilities. The LS attribute corresponds to expected normalized flows and takes positive values but is numerically close to zero for a majority of the links in the network. \( \hat{\omega}_{LS} \) and \( \hat{\omega}_{OL} \) indicate that the scales are inversely related to flow and number of outgoing links; links with more flow and more outgoing links have smaller variance of the error terms than links with less flow and fewer outgoing links.

It is not straightforward to analyze the resulting scale parameters based on \( \hat{\omega} \). We therefore provide two histograms in Figure 3.6 showing the distribution of \( \mu_k \) and \( \ln \phi_{ka} = \ln \frac{\mu_k}{\mu_k} \) over the links in the network for the NRL-LS model. The graph on the left shows that the values of \( \mu_k \) vary over the links in the network which ensures that IIA does not hold (the average value of \( \mu_k \) is 0.78). The peaks in the distribution are due to the attribute number of outgoing links \( OL(k) \) which take discrete values. We note that a few links have values larger than one: this is consistent with utility maximization and does not imply counter intuitive path probabilities. The graph on the right shows the distribution of \( \ln \phi_{ka} \) which is quite symmetric around 0 (the average value of \( \phi_{ka} \) is 1.03). The symmetry can be explained by the attribute \( OL(k) \). Consider the u-turn link \( a' \) of link \( a \in A(k) \). Since link \( k \) and \( a' \) have the same sink node we have
\( OL(k) = OL(a') \). For our data this results in values of \( \mu_k \) numerically close to \( \mu_{a'} \) and thus

\[
\phi_{ka} \phi_{aa'} = \frac{\mu_a \mu_{a'}}{\mu_k \mu_a} \approx 1
\]

or equivalently

\[
\ln \phi_{ka} + \ln \phi_{aa'} \approx 0.
\]

![Figure 3.6: Histogram of \( \mu_k \) and \( \ln \phi_{ka} \) for NRL-LS](image)

The LS attribute was designed to correct the utilities of overlapping paths in a way similar to the path size attribute. Moreover, if the values of these attributes are updated in case of a change in any attribute in the network, they relax the IIA property. Several studies in the literature (e.g. Bekhor et al., 2001, Frejinger and Bierlaire, 2007) report a better model fit and prediction results if these attributes are included in the deterministic utilities in addition to correlated random terms. It is also the case in this study: we observe a significant improvement in final log-likelihood values when we add the LS attribute (the likelihood ratio test are reported in Table 3.6, when applicable). The best model in terms of in-sample fit is NRL-LS. Since the scale parameters and the link size parameter are estimated off the same variation in the data, it is important to note that an identification issue may occur. It is however not the case for this data set.

Before comparing prediction results in the following section we make some remarks concerning the estimation. We use a basic trust region algorithm with the BHHH method for approximating the Hessian and the code is implemented in MATLAB (and available upon request). We use the iterative method with dynamic accuracy for the computation of the value functions (see Section 3.4). We note that if we use an initial vector as a solution of the system of linear equations, about 100 iterations is enough for a high precision \( \gamma' = 10^{-8} \) but we need about 200 iterations for the same precision when the initial vector is the unit vector (all the elements are equal to one). Moreover, using only 50 iterations in the beginning of the optimization (corresponding to a precision \( \gamma' \in [1, 10] \)) and switching to the high precision \( \gamma' = 10^{-8} \) when the norm of the
<table>
<thead>
<tr>
<th>Parameters</th>
<th>RL</th>
<th>NRL</th>
<th>RL-LS</th>
<th>NRL-LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\beta}_{TT}$</td>
<td>-2.494</td>
<td>-1.854</td>
<td>-3.060</td>
<td>-2.139</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.098</td>
<td>0.132</td>
<td>0.103</td>
<td>0.145</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-25.45</td>
<td>-14.05</td>
<td>-27.709</td>
<td>-14.75</td>
</tr>
<tr>
<td>$\hat{\beta}_{LT}$</td>
<td>-0.933</td>
<td>-0.679</td>
<td>-1.057</td>
<td>-0.748</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.030</td>
<td>0.043</td>
<td>0.029</td>
<td>0.047</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-31.10</td>
<td>-15.79</td>
<td>-36.448</td>
<td>-15.91</td>
</tr>
<tr>
<td>$\hat{\beta}_{LC}$</td>
<td>-0.411</td>
<td>-0.258</td>
<td>-0.353</td>
<td>-0.224</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.013</td>
<td>0.016</td>
<td>0.011</td>
<td>0.015</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-31.62</td>
<td>-16.13</td>
<td>-32.091</td>
<td>-14.93</td>
</tr>
<tr>
<td>$\hat{\beta}_{UT}$</td>
<td>-4.459</td>
<td>-3.340</td>
<td>-4.531</td>
<td>-3.301</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.114</td>
<td>0.200</td>
<td>0.126</td>
<td>0.207</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-39.11</td>
<td>-16.7</td>
<td>-35.960</td>
<td>-15.95</td>
</tr>
<tr>
<td>$\hat{\beta}_{LS}$</td>
<td>-</td>
<td>-</td>
<td>-0.227</td>
<td>-0.155</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>-</td>
<td>0.013</td>
<td>0.013</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>-</td>
<td>-17.462</td>
<td>-11.92</td>
</tr>
<tr>
<td>$\hat{\omega}_{TT}$</td>
<td>-</td>
<td>0.515</td>
<td>-</td>
<td>0.341</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.255</td>
<td>-</td>
<td>0.288</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>2.02</td>
<td>-</td>
<td>1.18</td>
</tr>
<tr>
<td>$\hat{\omega}_{LS}$</td>
<td>-</td>
<td>-0.674</td>
<td>-</td>
<td>-0.581</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.093</td>
<td>-</td>
<td>0.090</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>-7.25</td>
<td>-</td>
<td>-6.46</td>
</tr>
<tr>
<td>$\hat{\omega}_{OL}$</td>
<td>-</td>
<td>-0.086</td>
<td>-</td>
<td>-0.092</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.015</td>
<td>-</td>
<td>0.016</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>-5.73</td>
<td>-</td>
<td>-5.75</td>
</tr>
<tr>
<td>$LL(\hat{\beta})$</td>
<td>-6303.9</td>
<td>-6187.9</td>
<td>-6045.6</td>
<td>-5952.0</td>
</tr>
</tbody>
</table>

Table 3.5: Estimation results

<table>
<thead>
<tr>
<th>Models</th>
<th>$\chi^2$</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL &amp; NRL</td>
<td>232</td>
<td>5.11e-50</td>
</tr>
<tr>
<td>RL-LS &amp; NRL-LS</td>
<td>187.2</td>
<td>2.46e-40</td>
</tr>
<tr>
<td>NRL &amp; NRL-LS</td>
<td>471.8</td>
<td>1.30e-104</td>
</tr>
</tbody>
</table>

Table 3.6: Likelihood ratio test results

The gradient of the log-likelihood function is less than $10^{-3}$, we were able to double the speed of the estimation.

### 3.6.3 Prediction Results

In this section we focus on comparing the prediction performance of the different models. We use a cross validation approach where the sample of observations is divided into two sets by drawing observations at random with a fixed probability: one set is used for estimation (80% of the observations) and the other is used as holdout (20% of the observations) to evaluate the predicted probabilities by applying the estimated model. We generate 40 holdout samples of the
same size by reshuffling the real sample. The log-likelihood loss is used as the loss function to evaluate the prediction performance. More precisely, for each holdout sample $i$, $0 \leq i \leq 40$ we estimate the parameters $\hat{\beta}_i$ off the corresponding training sample and this vector of parameters is used to compute the test errors $err_i$

$$err_i = -\frac{1}{|PS_i|} \sum_{\sigma_j \in PS_i} \ln P(\sigma_j, \hat{\beta}_i)$$

where $PS_i$ is the size of prediction sample $i$. Then $err_i$ is a random variable that depends on the holdout sample $i$. In order to have unconditional test error values we compute the average of $err_i$ values over samples as follows

$$\overline{err}_p = \frac{1}{p} \sum_{i=1}^{p} err_i \quad \forall 1 \leq p \leq 40.$$ (3.22)

The values of $\overline{err}_p$, $1 \leq p \leq 40$ are plotted in Figure 3.7 and Table 3.7 reports the average of the test error values over 40 samples given by the RL, RL-LS, NRL, NRL-LS models. For each model the value of $\overline{err}_p$ becomes more stable as $p$ increases. The prediction results show that models including the LS attribute perform better than those without. The NRL-LS model has a significantly better fit and also a better prediction performance than RL-LS. We note that the differences between the models’ test error values are quite constant over the holdout samples. This is due to the fact that (i) the same holdout sample is used across models, and (ii) the number of observations used for estimation and the size of the holdout samples are large, so the parameter estimates are stable and so are the predicted log-likelihood values.

<table>
<thead>
<tr>
<th>RL</th>
<th>NRL</th>
<th>RL-LS</th>
<th>NRL-LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.392</td>
<td>3.336</td>
<td>3.252</td>
<td>3.204</td>
</tr>
</tbody>
</table>

Table 3.7: Average of test error values over 40 holdout samples

### 3.7 Conclusion

This paper has presented the NRL model which avoids the IIA property of the RL model by allowing scale parameters to be link specific while keeping the advantages of the RL model. We have proposed an efficient approach to estimate the model, solving the value functions using an iterative method with dynamic accuracy. Moreover, we have derived the gradient of the log-likelihood function which can be computed by solving systems of linear equations.

We have provided numerical results using real data. The parameter estimates are sensible and the NRL model has significantly better fit than the RL model. The LS attribute plays an important role and the best models including this attribute have significantly better model fit
than those without. We have also provided a cross-validation study suggesting that NRL-LS and NRL are better than the RL-LS and RL model, respectively.

In future research we plan to investigate further the importance of the LS attribute and its definition. Moreover, there are only few attributes available in the data set used in this paper. We would like to test the model on other data sets that allows us to test other possible functional forms of the scale parameters.

In this paper we use a unimodal network and observations of trips made by car. We emphasize that the model is not restricted to this type of network. More precisely, by adapting the state space, the model can be used in e.g. dynamic networks (state is time and location) and multimodal networks (state is location and mode) as long as link attributes are deterministic. The dynamic network is suitable for modeling congested networks, the RL model has been used for this purpose by Ramos et al. (2012). The challenge lies in the size of the state space, which is considerably larger than a static network since it is the number of links multiplied by the number of time intervals.

As a final remark we note that since the RL and NRL models are based on the universal choice set (including all path even those with loops), they avoid having to consider choice set formation. They can therefore be seen as alternatives to the approach proposed by Manski (1977). The RL and NRL may be relevant to other contexts than route choice where there is an issue associated with large choice sets.
Chapter 4

Integrating Correlation Structures for a Generalized Recursive Model

We build a general dynamic route choice model based on a dynamic discrete choice framework with correlated random terms. This model is more flexible, compared to other MEV-based recursive models, since it allows the utilities at each stage to be correlated by a network-based correlation structure. The estimation of the generalized model requires solving complicated dynamic programming problems. We deal with this challenge by proposing an innovative method to simplify the corresponding Bellman’s equation. We apply the new method on real data and show that a dynamic model with cross-nesting structures at choice stages can be quickly estimated and it outperforms the other recursive models in terms of in-sample and out-of-sample fit.

The resulting model generalizes the RL and the NRL model presented in Chapter 3. In order to deal with the estimation of this generalized model, we simplify Bellman’s equation so that it can be solved using a new artificial network and the value iteration with dynamic accuracy proposed in Chapter 3. The idea of this paper can be of significance in other contexts than route choice, as it can be used to simplify the estimation of dynamic discrete choice models with MEV random terms.

This work was presented at the 4rd Symposium of the European Association for Research in Transportation, DTU Transport, Copenhagen, Denmark, September 10-13, 2015. It is also available as CIRRELT (Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, Université de Montréal) technical report (Mai, 2015). The paper has been submitted for possible publication in Transportation Research Part B.
A Method of Integrating Correlation Structures for a Generalized Recursive Route Choice Model

Tien Mai

Abstract

We propose a model that generalizes existing recursive logit models in the literature, i.e., Fosgerau et al. (2013b), Mai et al. (2015c) by allowing the choice model at each stage to be any member of the network multivariate extreme value (network MEV) (Daly and Bierlaire, 2006). The estimation of the generalized model requires defining a contraction mapping and performing contraction iterations to solve Bellman’s equation. Given the fact that the contraction mapping is defined based on the choice probability generating functions (CPGF) (Fosgerau et al., 2013b) of the network MEV models, and these CPGFs are complicated, the generalized model becomes difficult to estimate. We deal with this challenge by proposing a novel method where the network of correlation structures and the structure parameters given by the network MEV models are integrated into the transport network. The approach allows to simplify the contraction mapping which makes it possible to estimate the model using revealed preference data.

We specify a recursive cross-nested logit (RCNL) model, a member of the generalized model, where the choice model at each stage is a cross-nested logit. We report estimation results and a cross-validation study based on a real network, and the results show that the RCNL model performs better than the other recursive models.

Keywords: recursive network MEV, contraction mapping, integrated network, recursive cross-nested, value iteration, maximum likelihood estimation, cross-validation.

4.1 Introduction

Given a transport network with links and nodes, and given a origin-destination pair, the route choice problem deals with identifying which route would be taken by a traveler. Discrete choice choice models are generally used in the context. There are two main issues associated with the use of discrete choice models, namely, choice sets of paths are unknown to the analyst, and path utilities may be correlated. We consider a generalized recursive route choice model where the choice model at each stage belongs to a family of multivariate extreme value (MEV) models. The model has the advantages from the existing recursive models (Fosgerau et al., 2013a, Mai et al., 2015c), as it can be consistently estimated and is easy for prediction without sampling of choice sets. It is, moreover, flexible in the sense that it allows to capture the correlation at

---

1Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
each choice stage by many convenient static discrete choice models, e.g., multinomial, nested
logit (Ben-Akiva, 1973), cross-nested logit (Vovsha and Bekhor, 1998) or multi-level cross-nested
models (Daly and Bierlaire, 2006). However, the generalized model is impractical to estimate
if using the estimation methods proposed for the other recursive models. We therefore propose
an innovative method that allows to integrate the correlation structure given by the model at
each stage into the transport network and simplify the estimation.

The recursive logit (RL) model proposed by Fosgerau et al. (2013a) can be consistently estimated
on real data without sampling any choice sets of paths. It is assumed that travelers choose states
(nodes or links) in a sequential manner. At each state they maximize the sum of the random
utility associated with a successor state (instantaneous utility) and the expected maximum
utility from the state to the destination (also known as the value functions). The random terms
of the instantaneous utilities are assumed to be independently and identically distributed (i.i.d.)
extreme value type I and the RL model is equivalent to a multinomial logit (MNL) model over
choice sets of all feasible path alternatives. The RL model hence inherits the independence of
irrelevant alternatives (IIA) property which is undesirable in a route choice setting (for instance
Mai et al., 2015d). Recently, Mai et al. (2015c) proposed the nested RL (NRL) model that
relaxes the IIA property of the RL model by assuming that scale parameters are link specific.
Both the RL and NRL models, however, assume that the choice at each stage are MNL, so the
correlation between the utilities of successor states cannot be captured.

In this paper, we consider a generalization of these models where the choice model at each stage
can be any member of the network MEV model (Daly and Bierlaire, 2006), e.g., the MNL, the
nested logit (Ben-Akiva, 1973) or cross-nested logit (Vovsha and Bekhor, 1998). This model,
called the recursive network MEV (RNMEV), is fully flexible in the sense that the cross-nested
logit can approximate any additive random utility model (for instance Fosgerau et al., 2013b).
However, the estimation of the generalized model leads to a complicated dynamic programming
(DP) problem, which is cumbersome to solve. We explain this challenge in more detail in the
following.

All the existing recursive route choice models are based on the dynamic discrete choice frame-
work proposed by Rust (1987), where the estimation requires solving a Bellman’s equation to
obtain the value functions. Rust (1987) shows that the value functions can be computed by
defining a contraction mapping \( V = T_\beta(V) \), where \( T_\beta \) is a contraction mapping associated with
parameters \( \beta \), and \( V \) is the vector of the value functions. The fixed point solution, or the value
functions, then can be obtained by using the method of successive approximations (or value
iteration) based on \( T_\beta \). In practice, this method needs to be formulated as matrix operations in
order to deal with problems with large number of states. In the RL and NRL models, due to
the simple choice probability generating functions (CPGF) (Fosgerau et al., 2013b) given by the
MNL model at the choice stages, the corresponding contraction mappings have simple forms
(even linear form in the RL model). Consequently, the corresponding contraction iterations can be formulated easily as matrix operations, and the RL and NRL models can be estimated quickly for real size networks. On the contrary, the CPGFs given by the network MEV model are complicated. They have closed forms but need to be computed recursively based on the network of correlation structure (see for instance Daly and Bierlaire, 2006). Hence, the contraction iterations given by the RNMEV model are not straightforward to formulate as matrix operations. This explains why the estimation of the RNMEV model is extremely expensive if using directly the standard approaches proposed in Fosgerau et al. (2013a), Mai et al. (2015c) and Rust (1987). In this context, we note that the generalized model also relaxes the well-known CLOGIT assumption from the Rust’s model (Rust, 1987), i.e., the assumption that the choice at each stage is MNL.

We propose an innovative method that allows us to simplify the contraction mapping given by the RNMEV and to quickly estimate the generalized model on real data. More precisely, we consider the networks of correlation structures given by the network MEV models (also known as MEV-networks) at the choice stages. We create a new artificial network by integrating the MEV-networks into the transport network and associate the states and arcs of the new network with new parameters and utilities based on those from the original network and the structure parameters of the network MEV models. We show similarities between the value functions given by the RNMEV model and those given by the NRL model on the integrated network. This method allows us to avoid recursively computing the CPGFs given by the network MEV models, and to use the contraction mapping from the NRL model to obtain the value functions of the RNMEV model. This approach, therefore, greatly simplifies the estimation of the RNMEV model.

We apply the new method on real data by specifying a recursive cross-nested logit (RCNL) model, a member of the RNMEV model, where the choice model at each stage is a cross-nested logit model (Vovsha and Bekhor, 1998). In the RCNL model, the variance-covariance matrix at each choice stage is no longer diagonal as in the RL and NRL models. This model therefore exhibits a more general correlation structure, compared to the other recursive models. We provide estimation and prediction results for a real network, and show that the RCNL model can be estimated in reasonable time while performing significantly better than the RL and NRL models in terms of in-sample and out-of-sample fit.

This paper makes two main contributions. First, we propose the method of integrating correlation structures to simplify the estimation of the RNMEV. Second, we propose the RCNL model that can flexibly capture the correlation structure at each choice stage, and we provide estimation and cross-validation results of the RCNL for a real network using real observations. Moreover, the estimation code for estimating the RCNL is available in an open source project and we share it freely upon request.
The paper is structured as follows. Section 4.2 introduces the RNMEV model and Section 4.3 presents the method of integrating correlation structure. Section 4.4 discusses in detail the maximum likelihood estimation. Section 4.5 presents the RCNL model. We provide the estimation results and a cross-validation study in Section 4.6, and finally, Section 4.7 concludes. The proofs of the theorems presented in this paper are provided in Appendices A.1 and A.2.

4.2 Generalized Recursive Route Choice Model

In the recursive route choice models proposed in Fosgerau et al. (2013a) and Mai et al. (2015c), the path choice problem is formulated as a sequence of link choices and modeled in a dynamic discrete choice framework. At each sink node of a link the decision maker chooses the utility-maximizing outgoing link with link utilities given by the instantaneous utility and the expected maximum utility to the destination. The random terms at each state are i.i.d. extreme value type I, so the choice model at each stage is MNL. In this section, we generalize the RL and NRL models by assuming the choice at each stage to be a network MEV model (Daly and Bierlaire, 2006). In order to better describe the generalized model, we consider the road network as a set of states and arcs connecting states. The states can be nodes in the network, or links as in Mai et al. (2015c) and Fosgerau et al. (2013a). The instantaneous utilities are defined for states conditional on other states and the path choice problem is formulated as a sequence of state choices, and there are states representing destinations in the road network.

A directed connected graph (not assumed acyclic) $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ is considered, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and arcs, respectively. For each state $k \in \mathcal{A}$, we denote the set of successor states of $k$ by $S(k)$ (if states are links, $S(k)$ is the set of the outgoing links from the sink node of $k$). Moreover, we associate an absorbing state with the destination of a given individual by extending the network with a dummy state $d$ that has no successor (see Figure 4.1). The set of all states is therefore $\tilde{\mathcal{S}} = \mathcal{S} \cup \{d\}$ and the corresponding deterministic utility is $v(d|k) = 0$ for all $k$ that connects to $d$.

$\begin{align*}
v(a_1|k) + \epsilon(a_1|k) & \quad V(a_1) \\
\cdots & \\
\cdots & \\
k & S(k) \\
\cdots & \\
a_2 & d
\end{align*}$

Figure 4.1: Illustration of notation
Given two states \( a, k \in \tilde{S} \) and individual \( n \), the following instantaneous utility is associated with state \( a \) conditional on \( k \)

\[
u^n(a|k; \beta) = v^n(a|k; \beta) + \epsilon(a|k; \beta) - \frac{\gamma}{\mu_k(\beta)}, \forall k \in S, a \in S(k),
\]

where \( \beta \) is a vector of parameters to be estimated, \( \gamma \) is the Euler’s constant and random terms \((\epsilon(a|k), a \in S(k))\) follow an MEV distribution, with the CPGF \( G_k \) of homogeneous degree \( \mu_k > 0 \) generated by the network MEV model. We note that the term \( \frac{\gamma}{\mu_k(\beta)} \) is used in order to ensure that the random term has zero mean. The deterministic term \( v^d(a|k), a \in S(k) \), is assumed negative for all states except for the dummy \( d \) that equals 0, i.e., \( v^d(d|k) = 0, \forall k \in S \). For notational simplicity, we omit from now on an index for individual \( n \) but note that the utilities can be individual specific. Given a state \( k \in S \), the next state is chosen by taking the maximum utility as

\[
\arg\max_{a \in S(k)} \left\{ v(a|k; \beta) + V^d(a; \beta) + \epsilon(a|k; \beta) - \frac{\gamma}{\mu_k(\beta)} \right\}, \forall k \in S,
\]

where \( V^d(a; \beta), \forall a \in \tilde{S} \), is the expected maximum utility (or value function) from the state \( a \) to the destination, which is recursively defined by Bellman’s equation

\[
V^d(k; \beta) = \mathbb{E} \left[ \max_{a \in S(k)} \left( v(a|k; \beta) + V^d(a; \beta) + \epsilon(a|k; \beta) - \frac{\gamma}{\mu_k(\beta)} \right) \right], \forall k \in S, \tag{4.1}
\]

and \( V^d(d) = 0 \). The superscript \( d \) indicates that the value functions are destination specific and they also depend on parameters \( \beta \). However, for notational simplicity we omit from now on \( \beta \) and superscript \( d \) from the value functions \( V() \) and the utilities \( v() \). According to McFadden (1978), Bellman’s equation can be written as

\[
V(k) = \frac{\ln G_k(\epsilon v(a|k) + V(a), a \in S(k))}{\mu_k}, \forall k \in S, \tag{4.2}
\]

and note that \( V(d) = 0 \). If we define a vector of size \(|\tilde{S}| \) (\(| \cdot | \) is the cardinality operator) with entries

\[
Y_k = e^{\mu_k V(k)}, \forall k \in \tilde{S},
\]

then the system in \( (4.1) \) can be written as

\[
Y_k = \begin{cases} 
G_k(\epsilon v(a|k) Y_a^{1/\mu_a}, a \in S(k)), & \text{if } k \in S \\
1 & \text{if } k = d
\end{cases}. \tag{4.3}
\]

Moreover, the probability of choosing state \( a \) given state \( k \) is given by the MEV model

\[
P(a|k) = \delta(a|k) \frac{y(a|k) \partial G_k}{\partial y(a|k)} (y(a'|k), a' \in S(k)) \frac{\partial G_k}{\mu_k G(y(a'|k), a' \in S(k))}, \forall k, a \in \tilde{S}, \tag{4.4}
\]
where \( y(a|k) = e^{y(a|k)}A^1/\mu_a, \forall a \in S(k) \). Note that we include \( \delta(a|k) \) that equals one if \( a \in S(k) \) and zero otherwise so that the probability is defined for all states \( a, k \in \tilde{S} \).

The probability of a path \( \sigma \) defined by a sequence of states \( \sigma = [k_0, k_1, \ldots, k_I] \) has a more complicated form than the ones given by the RL and NRL models. In general, it can be expressed as

\[
P(\sigma) = \prod_{i=0}^{I-1} P(k_{i+1}|k_i),
\]

in which \( P(k_{i+1}|k_i) \) can be computed using (4.4).

Now we turn our attention to the network MEV model at each stage. We assume that for each state \( k \in S \), the respective CPGF \( G_k(y) \) is generated by a network MEV model based on a cycle-free network of correlation structure \( G_k = (\mathcal{S}_k, \mathcal{A}_k, \mathcal{C}_k) \), where \( \mathcal{S}_k \) is the set of states, \( \mathcal{A}_k \) is the set of arcs and \( \mathcal{C}_k \) is the set of states that represent alternatives. Note that the set of states representing alternatives in this network MEV model is also the set of next states from \( k \) that is \( S(k) \). So each state \( i \in \mathcal{C}_k \) corresponds to only one state \( a \in S(k) \) and vice versa. Figure 4.2 illustrates a network of correlation structure at state \( k \). Each arc \((i, j) \in \mathcal{A}_k \) is associated with a positive parameter \( \alpha^k_{ij} \) and each state \( i \in \mathcal{S}_k \) is associated with a positive scales \( \xi^k_i \). The CPGF (with respect to a vector of parameters \( y \)) associated with each state in \( \mathcal{S}_k \) are defined as

\[
G^i_k(y) = y_i^{\xi^k_i}, \quad i \in \mathcal{C}_k,
\]

and

\[
G^i_k(y) = \sum_{j \in \mathcal{S}_k(i)} \alpha^k_{ij} (G^j_k(y))^{\xi^k_i/\xi^k_j}, \forall i \in \mathcal{S}_k \setminus \mathcal{C}_k,
\]

where \( \mathcal{S}_k(i) \) is the set of the successors of state \( i \) in network \( G_k \).

We obtain the CPGF \( G_k(y) \) as \( G_k(y) = G^r_k(y) \), where \( r \) is the root of network \( G_k \). Daly and Bierlaire (2006) show that \( G_k(y) = G^r_k(y) \) is a \( \xi^k \)-MEV CPGF and \( \mu_k \) is the homogeneous degree of \( G_k(y) \), so \( \xi^k = \mu_k \). Moreover, Daly and Bierlaire (2006) show that the probability \( P_k(i|C_k; y) \)
of choosing alternative $i \in C_k$ can be expressed based on the CPGFs defined in (4.6) and (4.7) as

$$P_k(i|C_k; y) = \sum_{[j_0, \ldots, j_I] \in \Omega^k(i)} \prod_{t=0}^{I-1} \alpha_{j_tj_{t+1}}^k \left( G_k^{j_{t+1}}(y) \right)^{e_k} G_k^{j_I}(y),$$

(4.8)

where $\Omega^k(i)$ is the set of all paths connecting the root $r$ and $i$. A path is defined by a sequence of states $[j_0, \ldots, j_I]$ such that $j_{t+1} \in S_k(j_t)$, $\forall t = 0, \ldots, I-1$, where $j_0$ is the root $r$ and $j_I$ represents alternative $i$. If we denote $y_k$ a vector of size $|S(k)|$ with entries $(y_k)_a = y(a|k) = e^{v(a|k)} Y^1_a / \mu_a$, $\forall a \in S(k)$, then according to (4.3) we have $Y_k = G_k(y_k)$ and the probability $P(a|k)$ for a state $a \in S(k)$ can be computed by using (4.8). In other words, $P(a|k) = P_k(i_a|C_k; y_k)$, where $i_a$ is a state in $C_k$ corresponding to state $a \in S(k)$.

The network MEV model generalizes many MEV models in the literature and examples are the MNL, the nested logit (Ben-Akiva, 1973), the paired combinatorial logit (Koppelman and Wen, 2000), the generalized nested logit (Wen and Koppelman, 2001), the ordered MEV model (Small, 1987), the cross-nested logit model (Vovsha and Bekhor, 1998) and the GenL model (Swait, 2001) models. So the RNMEV model allows to capture the correlation at each choice stage by many convenient discrete choice models. Indeed, if all the $G_k$ functions, $\forall k \in S$, refer to the MNL model, then the RNMEV becomes the NRL model.

The estimation of the RNMEV model requires solving Bellman’s equation in (4.2). This can be done by defining a contraction mapping $V = T_\beta(V)$ (Rust, 1987), and applying the method of successive approximations (value iteration), i.e., performing $V^{(t+1)} = T_\beta(V^{(t)})$ iteratively until the fixed point solution is found. The contraction mapping $T_\beta$ is defined based on CPGFs $G_k(\cdot)$, $\forall k \in S$, where the computation of CPGF $G_k(\cdot)$ requires performing the recursive equations in (4.7) and (4.6), leading to the fact that $T_\beta$ is complicated. Given the fact that there are too many $G_k(\cdot)$ to be computed, the computation of the value functions becomes impractical if using standard approaches. The gradients of the value functions are cumbersome to evaluate as well. In the next section, we show how to simplify the computation of the value functions as well as the choice probabilities by integrating the networks of correlation structures $G_k$, $\forall k \in S$, into the road network $\mathcal{G}$.

### 4.3 Method of Integrating Correlation Structures

In this section, we show how to integrate the MEV-networks at choice stages into the transport network in order to simplify the contraction mapping given by the RNMEV model. More precisely, we first introduce a method to integrate the networks of correlation structures $G_k$, $\forall k \in S$, into the road network $\mathcal{G} = (\mathcal{S}, \mathcal{A})$, second we show similarities between the RNMEV
and the NRL model on the integrated network. These similarities allow us to use the contraction mapping given by the NRL to quickly compute the value functions of the RNMEV model.

### 4.3.1 Integrated Network

Given a state $k \in S$, the choice at $k$ is a network MEV model based on a network or correlation structure $G_k = (S_k, A_k, C_k)$. As mentioned earlier, the set of states $C_k$ (representing alternatives) is also the set of next states from $k$, i.e., $S(k)$. So in order to integrate $G_k$ to the road network we assume that $C_k \equiv S(k)$ and $k \equiv r$. Hence, the integrated network $G^* = (S^*, A^*)$ can be created by adding all sets $S_k$ and $A_k$, $\forall k \in S$, to the set of states $S$ and set of arcs $A$. In other words

$$S^* = \bigcup_{k \in S} S_k \text{ and } A^* = \bigcup_{k \in S} A_k.$$  \hspace{1cm} (4.9)

Basically, the new network $G^*$ is created by adding new states to $G$. For each $k \in S$, we add new states such that the subnetwork between $k$ and all states $a \in S(k)$ is similar to the network of correlation structure $G_k$. We also denote $S^*(k)$ as the set of successor states of state $k$ in network $G^*$.

We note that, due to the properties of the MEV-networks (for instance Daly and Bierlaire, 2006), $G^*$ remains connected and there are paths connecting between any two states $k, a \in S, a \in S(k)$. For the sake of illustration, we show in Figure 4.3 a small example where state $k$ has three successors $a_1, a_2$ and $a_3$ as illustrated in the left part of Figure 4.3. The network of correlation structure $G_k$ is given in the middle of the figure and in the right we show the integrated network $G^*$ at state $k$.

![Figure 4.3: Original and integrated network](image)

We introduce the following proposition related to the properties of the network $G^*$, which are easy to verify.

**Proposition 4.1.** *Network $G^*$ has the following properties*
(i) Given a state \( i \in S^* \), there is a state \( k \in S \) such that \( i \in S_k \).

(ii) Given a state \( i \in S^* \), if \( i \notin \tilde{S} \) then there exists only one state \( k \in S \) such that \( i \in S_k \).

(iii) Given a state \( i \in S_k \), if \( i \in \tilde{S} \) then \( i = k \) or \( i \in S(k) \).

(iv) Given a state \( i \in S_k \), if \( i \notin C_k \) then \( S_k(i) = S^*(i) \).

(v) \( \tilde{S} \cap S_k = \{ k \} \cup S(k) \) and \( |S^*| = |\tilde{S}| + \sum_{k \in S} (|S_k| - |S(k)| - 1) \).

(vi) \( A_k \cap A_h = \emptyset \) \( \forall k, h \in S, \ k \neq h \) and \( |A^*| = \sum_{k \in S} |A_k| \).

Proof. (i), (iii), (v) and (vi) are obviously verified by the definition in (4.9). We have the fact that given a state \( i \), if \( i \in S_k \cap S_h \) (with \( k, h \in S, k \neq h \)) then \( i \in \tilde{S} \). So if state \( i \notin \tilde{S} \) then there is only one state \( k \in S \) such that \( i \in S_k \). This proves (ii).

For proving (iv) we note that given \( i \in S_k \) \( \setminus \) \( C_k \) and if \( j \in S_k(i) \) for a given \( j \in S^* \) then \((i, j) \in A_k\), meaning that \((i, j) \in A^*\) by (4.9), or equivalently \( j \in S^*(i) \). Moreover, if \( j \in S^*(i) \) then \((i, j) \in A^*\). From (vi) there is only a state \( k' \in S \) such that \((i, j) \in A_{k'}\). Moreover, (ii) leads to the fact that \( k = k' \), so \((i, j) \in A_k \) or \( j \in S_k(i) \). Finally, we obtain \( S_k(i) \subseteq S^*(i) \) and \( S^*(i) \subset S_k(i) \). Hence, \( S_k(i) = S^*(i) \) and (iv) is proved.

Now we integrate the structural parameters given by the network MEV models into the integrated network. We consider network \( G^* = (S^*, A^*) \). We note that \( d \in S^* \) and \( S^*(d) = \emptyset \). We associate each state \( i \in S^* \) a positive parameter \( \mu_i^* \) as

\[
\mu_i^* = \begin{cases} 
\mu_i & \text{if } i \in \tilde{S} \\
\xi_i^k & \text{if } i \notin \tilde{S}, i \in S_k, k \in S.
\end{cases}
\]  

(4.10)

Recall \( \tilde{S} = S \cup \{ d \} \). Note that due to Proposition 4.1(ii), for \( i \notin \tilde{S} \), there is only one set \( S_k \) such that \( i \in S_k \), so there is only one value \( \xi_i^k \) such that \( i \in S_k, k \in S \).

For each arc \((i, j) \in A_k, k \in S\), the following deterministic utility is associated with state \( j \) conditional on \( i \)

\[
v^*(j|i) = \begin{cases} 
\ln(\alpha_i^{kj})/\mu_i^* & \text{if } j \notin S(k) \\
\ln(\alpha_i^{kj})/\mu_i^* + v(j|k) & \text{if } j \in S(k),
\end{cases}
\]

(4.11)

here we recall that \( v(j|k), k \in S, j \in S(k) \), is a deterministic utility associated with state \( j \) conditional on \( k \) and \( \alpha_i^{kj} \) are positive parameters of the network MEV model at state \( k \).
4.3.2 NRL Model on the Integrated Network

Now we apply the NRL model (Mai et al., 2015c) to network \( \mathcal{G}^* \). Given two states \( k, a \in \mathcal{S}^* \), \( a \in \mathcal{S}^*(k) \), the following instantaneous utility associated with state \( a \) given \( k \)

\[
u^*(a|k) = v^*(a|k) + \frac{\epsilon(a) - \gamma}{\mu_k^*},
\]

(4.12)

where \( \epsilon(a) \) are i.i.d extreme value type I and \( v^*(a|k), \mu_k^* \) are defined in (4.11) and (4.10). We notice that, in order to be consistent with the RNMEV model, the scales of the random terms in the NRL model are \( \frac{1}{\mu_k^*} \) instead of \( \mu_k^* \) in Mai et al. (2015c).

The expected maximum utility from the sink node of \( k, k \in \mathcal{S}^* \), to the destination is the value function \( V^*(k) \) that is recursively defined by Bellman’s equation

\[
V^*(k) = \mathbb{E}\left[ \max_{a \in \mathcal{S}^*(k)} \left\{ v^*(a|k) + V^*(a) + \frac{\epsilon(a) - \gamma}{\mu_k^*} \right\} \right], \tag{4.13}
\]

or equivalently (by the logsum)

\[
\mu_k^* V^*(k) = \ln \left( \sum_{a \in \mathcal{S}^*(k)} e^{\mu_k^* \left( v^*(a|k) + V^*(a) \right)} \right), \quad \forall k \in \mathcal{S}^* \setminus \{d\}, \tag{4.14}
\]

and note that \( V^*(d) = 0 \). If we define a vector \( Y^* \) of size \( |\mathcal{S}^*| \) with entries \( Y_k^* = e^{\mu_k^* V^*(k)} \), \( \forall k \in \mathcal{S}^* \), then Bellman’s equation becomes

\[
Y_k^* = \begin{cases} 
\sum_{a \in \mathcal{S}^*(k)} e^{\mu_k^* v^*(a|k)} (Y_a^*)^{\mu_k^* / \mu_k^*} & \text{if } k \neq d \\
1 & \text{if } k = d 
\end{cases}, \tag{4.15}
\]

Moreover, the probability of choosing state \( a \) given \( k \) is given by the MNL as

\[
P^*(a|k) = \delta^*(a|k) \frac{e^{\mu_k^* v^*(a|k)} (Y_a^*)^{\mu_k^* / \mu_k^*}}{Y_k^*}, \quad \forall k, a \in \mathcal{S}^*, \tag{4.16}
\]

where \( \delta^*(a|k) \) equals one if \( a \in \mathcal{S}^*(k) \) and zero otherwise so that the probability is defined for all \( k, a \in \mathcal{S}^* \). We note that the system in (4.15) is non-linear but can be solved quickly for a large network using the approach proposed in Mai et al. (2015c), namely a value iteration method with dynamic accuracy.
4.3.3 Similarities between the NRL on Network $G^*$ and RNMEV on Network $G$

This section presents similarities between the value functions and choice probabilities given by the NRL on network $G^*$ and those given by the RNMEV on network $G$. We first introduce a theorem related to the value functions.

**Theorem 4.2.** If vector $Y^*$ is a solution to the non-linear system (4.15) then

$$Y^*_k = G_k(e^{v(a|k)}(Y^*_a)^{1/\mu_a}, a \in S(k)), \forall k \in S.$$  \hfill (4.17)

In other words, $Y_k = Y^*_k, \forall k \in \tilde{S}$, is a solution to Bellman’s equation of the RNMEV given by (4.3).

The next theorem shows that the state choice probabilities under the RNMEV model can be expressed via the probabilities given by the NRL model on network $G^*$.

**Theorem 4.3.** If vector $Y^*$ is a solution to the non-linear system (4.15), and if $Y_k = Y^*_k$, $\forall k \in \tilde{S}$, then

$$P(a|k) = \sum_{[a_0, \ldots, a_I] \in \Omega^k(a)} \prod_{t=0}^{I-1} P^*(a_{t+1}|a_t), \forall k \in S, a \in S(k).$$ \hfill (4.18)

We recall that $\Omega^k(a)$ is the set of sequences of states connecting $k$ and $a$: $[k = a_0, \ldots, a_I = a]$ such that $a_t \in S_k(a_{t-1}), \forall t = 1, \ldots, I$, and $a_{t+1} \in S_k(a_t), \forall t = 0, \ldots I - 1$.

We provide the proofs of the two theorems in Appendixes 4.2 and 4.3. These theorems indicate that the value functions and choice probabilities in the RNMEV model can be computed by using the respective values from the NRL model applying on network $G^*$. This means that the methods proposed in Mai et al. (2015c) can be used to estimate the RNMEV model. In the next section we discuss in detail the maximum likelihood estimation.

4.4 Maximum Likelihood Estimation

The nested fixed point algorithm (Rust, 1987) is often used to estimate a dynamic discrete choice model. This algorithm combines an outer iterative non-linear optimization algorithm for searching over the parameter space with an inner algorithm for solving Bellman’s equation to obtain the value functions. We have shown that the computation of the value functions and choice probabilities given by the RNMEV model can be simplified by applying the NRL model to the integrated network. The NRL model can be estimated efficiently using the methods in Mai et al. (2015c), namely, estimating the value functions with dynamic accuracy, and computing the
gradients of the log-likelihood function by solving systems of linear equations. In the following we briefly describe the computations of these values in the NRL model based on the integrated network.

### 4.4.1 Computation of the Value Functions

The main challenge associated with the NRL model is to solve the large-scale system of nonlinear to obtain the value functions. Similarity to Mai et al. (2015c), we define a matrix $M^{\ast}(\left|S^{\ast}\right| \times \left|S^{\ast}\right|)$ with entries

$$M_{ka}^{\ast} = \delta^{\ast}(a|k)e^{\mu^{\ast}\nu^{\ast}(a|k)}, \quad \forall k, a \in S^{\ast},$$

(4.19)

and a matrix $X$ of size $|S^{\ast}| \times |S^{\ast}|$ with entries

$$X(Y^{\ast})_{ka} = (Y_{a}^{\ast})^{\mu^{\ast}/\mu^{\ast} a}, \quad \forall k, a \in S^{\ast}$$

(4.20)

and $b$ is a vector of size $|S^{\ast}|$ with zero values for all states except for the dummy state $d$ with a value of 1. Bellman’s equation in (4.15) can be written in a matrix form as

$$Y^{\ast} = [M^{\ast} \circ X(Y^{\ast})]e + b,$$

(4.21)

where $e$ is a vector of size $(|S^{\ast}|)$ with value one for all states, and $\circ$ is the element-by-element product. A value iteration method can be used to solve this system, i.e., we start with an initial vector $(Y^{\ast})^{0}$ and then for each iteration $i$ we compute a new vector $(Y^{\ast})^{i+1} \leftarrow [M^{\ast} \circ X((Y^{\ast})^{i})]e + b$, and iterate until a fixed point solution is found using $||(Y^{\ast})^{i+1} - (Y^{\ast})^{i}|| \leq \gamma$, for a given threshold $\gamma > 0$ as stopping criteria. Mai et al. (2015c) show that the value iteration can be improved by using dynamic accuracy. The choice of initial vector is also important for the rate of convergence. Mai et al. (2015c) use the solution to the system of linear equations corresponding to the RL model (Fosgerau et al., 2013a) which is fast to compute. This choice can be improved by taking into account the solution of the previous iteration of the outer optimization algorithm (the algorithm for searching over the parameters space) and using a switching approach to select the best initial vector. More precisely, at iteration $t - 1$ of the outer algorithm, we suppose that the fixed point solution is $(Y^{\ast})^{t-1}$. At the next iteration $t$, we suppose $(Y^{\ast})^{0}$ is the fixed point solution of the RL model. For solving the value functions, the initial vector for the inner algorithm can be chosen by considering

$$err = ||(Y^{\ast})^{0} - [M^{\ast} \circ X((Y^{\ast})^{0})]e + b|| - ||(Y^{\ast})^{t-1} - [M^{\ast} \circ X((Y^{\ast})^{t-1})]e + b||.$$ 

If $err < 0$ than vector $(Y^{\ast})^{0}$ is chosen, otherwise we select $(Y^{\ast})^{t-1}$. This switching approach allows to select the better initial vector (closer to the fixed point solution).
4.4.2 Maximum Likelihood Estimation

Now we derive the log-likelihood (LL) function of the RNMEV model. The LL function defined over the set of path observations \( n = 1, \ldots, N \) is

\[
LL(\beta) = \sum_{n=1}^{N} \ln P(\sigma_n, \beta) = \sum_{n=1}^{N} \sum_{t=0}^{L_n} \ln P(k_{t+1}^n | k_t^n).
\]

(4.22)

We note that the path observations are defined based on states in network \( G \). Each probability \( P(k_{t+1}^n | k_t^n) \), \( k_{t+1}^n, k_t^n \in S \), can be computed using the results of Theorems 4.2 and 4.3.

For the maximum likelihood estimation, the network \( G_k, \forall k \in S \), generates MEV models with many parameters. That is \( \xi_k^i \) and \( \alpha_{ij}^k \), \( \forall i, j \in S_k \), need to be satisfied in order to ensure that the choice at state \( k \) is consistent with McFadden’s MEV theory. Based on the definition in (4.10) the constraints can be written as

\[
\mu_k^i \leq \mu_k^j, \forall i, j \in S^*, j \in S^*(i) \backslash \tilde{S}.
\]

Moreover, as suggested by Daly and Bierlaire (2006), a normalization for parameters \( \alpha_{ij}^k \) for the network MEV model at state \( k \) would require

\[
\sum_{i \in S^*} (\alpha_{ij}^k) \xi_i^k / \xi_j^k = 1, \forall j \in S_k, k \in S.
\]

This normalization, however, remains to be analyzed further.

Efficient nonlinear techniques for the problem require analytical derivatives of the LL function. The gradients of the LL function are complicated, but can be easily derived based on (4.22), (4.3) and (4.16). They require the gradients of \( Y^* \), \( \forall k \in S^* \), which are given by

\[
\frac{\partial Y_k^*}{\partial \beta_i} = Y_k^* \left( \frac{\partial \mu_k^*}{\partial \beta_i} V^*(k) + \frac{\partial V^*(k)}{\partial \beta_i} \mu_k^* \right), \forall k \in S^*,
\]

(4.23)

and

\[
\frac{\partial V^*}{\partial \beta_i} = (I - H)^{-1} (L^i e + h),
\]

(4.24)

where \( \beta_i \) is a parameter, \( L^i, H \) are two matrices of size \( |S^*| \times |S^*| \) and \( h \) is a vector of size \( |S^*| \) with entries

\[
L_{ka}^i = \frac{1}{\mu_k^a} \frac{\partial M_{ka}^* (Y_a^*)^{\phi_k a}}{\partial \beta_i} + \frac{1}{\mu_k^a} M_{ka}^* \ln(Y_a^*) \frac{(Y_a^*)^{\phi_k a}}{\mu_k^a} \frac{\partial \phi_k a}{\partial \beta_i} + M_{ka}^* \ln(Y_a^*) \frac{(Y_a^*)^{\phi_k a}}{(Y_k^*)} \frac{\partial \mu_k^a}{(\mu_k^a)^2 \partial \beta_i},
\]
and
\[ H_{ka} = M_{ka}^* \frac{(Y_k^*)^\phi_{ka}}{(Y_k^*)} \] and \( h_k = -\frac{\partial \mu_k^*}{(\mu^*)^2} \partial \beta_i \ln(Y_k^*) \) and \( \phi_{ka}^* = \mu_k^*/\mu_a^* \).

Mai et al. (2015c) suggest that deriving the gradients based on \( V^* \) is better than based on \( Y^* \) for numerical reasons. The system of linear equations in (4.24) can be solved quickly for real size networks.

4.5 Recursive Cross-nested Logit Model

The cross-nested logit (CNL) model is an instance of the network MEV model that allows each alternative to belong several different nests. It has been mentioned for the first time by Vovsha (1997) in the context of a mode choice survey in Israel. Papola (2004) has shown that a specific CNL model can be derived for any given homoscedastic variance-covariance matrix. Fosgerau et al. (2013b) show that any additive random utility model can be approximated by the cross-nested logit model. Therefore, the CNL model, with closed forms for the choice probabilities, becomes a serious competitor for the probit model.

In this section we present the RCNL model, which is an instance of the RNMEV model where the choice at each state is a CNL model based on the formulation proposed by Ben-Akiva and Bierlaire (1999a). In this setting, the CNL model at state \( k \) is a network MEV model given by a network of correlation structure \( \mathcal{G}^k \) where the corresponding CPGF \( G_k(y) \) is

\[
G_k(y) = \sum_{m \in S_k(r)} \left( \sum_{a \in S_k(m)} \alpha_{ma}^k \gamma_a^k \right) \zeta_r^k / \zeta_m^k, \tag{4.25}
\]

(recall that \( r \) is the root of the network MEV model at state \( k \)). We remark that \( S_k(r) \) is also the set of nests. For each state \( a \in S(k) \) we define \( z(a|k) = v(a|k) + V(a) \) and note that a next state is chosen by maximizing the sum of \( z(a|k) \) and the random term \( \epsilon(a|k) \), \( \forall a \in S(k) \). In order to model the correlation between the successor states \( a \in S(k) \), we define a CNL model where each pair of states belongs to only one nest, and each nest contains only one pair of states. The number of pairs in the set \( S(k) \) is \( \frac{1}{2} |S(k)|.(|S(k)| - 1) \), so it is also the number of nests at choice stage \( k \). Figure 4.4 illustrates an example where there are three successor states from \( k \). Accordingly, there are three nests \( m_{12}, m_{13} \) and \( m_{23} \). Two states \( a_i \) and \( a_j \) belong to nest \( m_{ij} \), \( \forall i, j = 1, 2, 3 \).

Given this correlation structure, based on Papola (2004), the correlation between two given states \( a_i, a_j \in S(k) \) can be approximated as

\[
\hat{\text{Corr}}(z(a_i|k), z(a_j|k)) = \sum_{m \in S_k(r)} (\alpha_{mi}^k \alpha_{mj}^k)^{0.5} \zeta_r^k / \zeta_m^k \left( 1 - \left( \frac{\zeta_r^k}{\zeta_m^k} \right)^2 \right). \tag{4.26}
\]
Since there is only one nest $m_{ij}$ that both $a_i$ and $a_j$ belong to, so

$$\hat{\text{Corr}} (z(a_i|k), z(a_j|k)) = (\alpha_{m_{ij}a_i}^k \alpha_{m_{ij}a_j}^k)^{0.5 \xi_r^k / \xi_{m_{ij}}^k} \left(1 - \left(\frac{\xi_r^k}{\xi_{m_{ij}}^k}\right)^2\right). \quad (4.27)$$

Hence, the correlation between $a_i, a_j \in S(k)$ can be modeled using the parameters $\alpha_{m_{ij}a_i}^k, \alpha_{m_{ij}a_j}^k, \xi_r^k$ and $\xi_{m_{ij}}^k$. It is important to note that if the choice model at state $k$ is MNL, then for any two states $a_i, a_j \in S(k), a_i \neq a_j$, we have $\text{Corr} (z(a_i|k), z(a_j|k)) = 0$, and the variance-covariance matrix is diagonal. So, the RCNL model allows to exhibit a more general correlation structure at each choice stage, compared to the NRL and RL models.

Abbé et al. (2007) note that Papola’s approximation in (4.27) can overestimate the correlation in some cases and bias the choice probabilities provided by the CNL model. However, they also comment that these biases do not seem to be large in their examples.

For the estimation, the RCNL model is a member of the RNMEV model, so this can be estimated by applying the method presented in Section 4.3, i.e., estimating the NRL model using the integrated network $G^*$ and using Theorems 4.2 and 4.3. According to Papola (2004), a normalization required for the CNL model given by (4.25) is

$$\sum_{m \in S_k(r)} (\alpha_{ma}^k)^{\xi_r^k / \xi_m^k} = 1. \quad (4.28)$$

Finally, the integrated network has larger state space, compared to the original (i.e. $G$). Based on Proposition 4.1, the numbers of states and arcs in the integrated network are

$$|S^*| = |\tilde{S}| + \frac{1}{2} \sum_{k \in S} (|S(k)| - 1) |S(k)|,$$
and
\[ |\mathcal{A}^*| = \frac{3}{2} \times \sum_{k \in \mathcal{S}} (|S(k)| - 1) |S(k)|. \]

4.6 Application

We illustrate the proposed methods and models by using the same data used in Fosgerau et al. (2013a), Mai et al. (2015c), which has been collected in Borlänge, Sweden. The network is composed of 3077 nodes and 7459 links and is uncongested so travel times can be assumed static and deterministic. The observations consist of 1832 trips corresponding to simple paths with a minimum of five links. Moreover, there are 466 destinations, 1420 different origin-destination (OD) pairs and more than 37,000 link choices in this sample. We present estimation and prediction results for the RCNL model. For the sake of comparison we include the results from the NRL model (Mai et al., 2015c). For the other recursive model, i.e., the RL model, reader can consult Mai et al. (2015c) for details.

4.6.1 Model Specification

We use the same attributes as Mai et al. (2015c) for the instantaneous utilities. We note that the other recursive route choice models in Mai et al. (2015c) and Fosgerau et al. (2013a) define the models and utilities based on links in the network to capture turn attributes. Since the models presented in this paper are based on a network of states, we then define the utility specifications based on states and note that each state refers to a link in the real transport network.

Five attributes are considered: travel time \( TT(a) \) of action \( a \), a left turn dummy \( LT(a|k) \) that equals one if the turn angle from \( k \) to \( a \) is larger than 40 degrees and less than 177 degrees, a u-turn dummy \( UT(a|k) \) that equals one if the turn angle is larger than 177, and a state constant \( LC(a) \). The fifth attribute is \( LS(a) \) (for a detailed description see Fosgerau et al., 2013a) and it has been computed using a linear in parameters formulation of the aforementioned four attributes using parameters \( \tilde{\beta}_{TT} = -2.5, \tilde{\beta}_{LT} = -1, \tilde{\beta}_{LC} = 0.4, \tilde{\beta}_{UT} = -4 \).

We specify the deterministic utilities for different model specifications with respect to state \( a \) given state \( k, k \in \mathcal{S}, a \in S(k) \)

\[
v^{\text{NRL}}(a|k; \beta) = v^{\text{RCNL}}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k),
\]

\[
v^{\text{NRL-LS}}(a|k; \beta) = v^{\text{RCNL-LS}}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k) + \beta_{LS}LS(a).
\]
In the NRL model in Mai et al. (2015c), the scale of the random terms are defined as exponential functions of the model parameters. Mai et al. (2015c) use the travel time, link size and number of successor states to define the scale as

\[ \mu_{NRL}^k(\omega) = e^{\omega_{TT}(k) + \omega_{LS}(k) + \omega_{OL}(k)}, \forall k \in \tilde{S}, \]

where \( OL(k) \) is the number of successor states from \( k \) i.e. \( OL(k) = |S(k)| \) (it is also the number of outgoing links from the sink node of a link in the road network). Note that this NRL is based on network \( G \) and differs from the NRL model based on the integrated network \( G^* \). The latter is used for the estimation of the RCNL model.

The CNL model at each state includes several structure parameters. Our network contains more than 7000 states/links, leading to more than 7000 CNL models. So it is not possible to estimate all the parameters. Similar to Mai et al. (2015c) we define parameters \( \xi_{kr} \) and \( \xi_{km} \), \( \forall m \in S_k(r) \) as exponential functions of the respective state attributes. More precisely, since at choice stage \( k \), the root \( r \) of the network MEV model is identical to state \( k \), so \( \xi_{kr} \) is defined as

\[ \xi_{kr}(\omega) = e^{\omega_{TT}(k) + \omega_{LS}(k) + \omega_{OL}(k)}, \forall k \in S. \] (4.29)

Indeed, by definition \( \xi_{kr}(\omega) > 0, \forall \omega \). Moreover, (4.27) suggests that the correlation between two given states \( a_i, a_j \) depends on the ratio \( \xi_{mij}^{k}/\xi_{kr}^{k} \). We therefore define this fraction as an exponential function of the attributes associated with states \( a_i \) and \( a_j \) as

\[ \frac{\xi_{mij}^{k}}{\xi_{kr}^{k}} = e^{\lambda_{TT}(TT(a_i) + TT(a_j)) + \lambda_{LS}(LS(a_i) + LS(a_j)) + \lambda_{OL}(OL(a_i) + OL(a_j))}, \] (4.30)

or equivalently the scale \( \xi_{mij}^{k} \) associated with nest \( m_{ij} \) is defined as

\[ \xi_{mij}^{k} = \xi_{kr}^{k} e^{\lambda_{TT}(TT(a_i) + TT(a_j)) + \lambda_{LS}(LS(a_i) + LS(a_j)) + \lambda_{OL}(OL(a_i) + OL(a_j))}, \] (4.31)

where \( \xi_{kr}^{k} \) is defined in (4.29). The CNL model requires constraints on the scale parameters which are \( \xi_{mij}^{k} \geq \xi_{kr}^{k}, \forall m_{ij} \in S_k(r) \). We therefore impose these constraints by restricting the parameter \( \lambda \) to being positive, i.e., \( \lambda_{TT}, \lambda_{LS}, \lambda_{OL} \geq 0 \).

In the CNL model at state \( k \), parameter \( \alpha_{ma}^{k} \) reflects the level of membership of alternative \( a \) to nest \( m \). Indeed, it is impossible to estimate all the parameters \( \alpha \) in the network. We therefore assume that each state \( a \in S(k) \), the levels of \( a \) to all the nests \( m \) are equal. Based on the normalization in (4.28) and as the number of nests that each node \( a \in S(k) \) belong to is \( |S(k)| - 1 \), the parameters are specified as

\[ \alpha_{ma}^{k} = \left( \frac{1}{|S(k)| - 1} \right) \frac{\xi_{m}^{k}/\xi_{kr}^{k}}{\xi_{m}^{k}/\xi_{kr}^{k}}. \] (4.32)
Hence, the correlation between two states \(a_i, a_j \in S(k)\) can be approximated as

\[
\overline{\text{Corr}}(z(a_i|k), z(a_j|k)) = \frac{1}{|S(k)| - 1} \left(1 - \left(\frac{\xi^k_r}{\xi^k_{mij}}\right)^2\right),
\]

where \(\xi^k_r\) is defined in (4.30). In summary, the instantaneous utilities are

\[
\begin{align*}
    u^{\text{NRL}}(a|k; \beta, \omega) &= v^{\text{NRL}}(a|k; \beta) + \frac{1}{\mu_k^{\text{NRL}}(\omega)} (\epsilon(a) - \gamma), \\
    u^{\text{NRL-LS}}(a|k; \beta, \omega) &= v^{\text{NRL-LS}}(a|k; \beta) + \frac{1}{\mu_k^{\text{NRL}}(\omega)} (\epsilon(a) - \gamma), \\
    u^{\text{RCNL}}(a|k; \beta, \omega, \lambda) &= v^{\text{RCNL}}(a|k; \beta) + \epsilon(a|k; \omega, \lambda), \\
    u^{\text{RCNL-LS}}(a|k; \beta, \omega, \lambda) &= v^{\text{RCNL-LS}}(a|k; \beta) + \epsilon(a|k; \omega, \lambda),
\end{align*}
\]

where \(\epsilon(a)\) is i.i.d extreme value type I and \(\epsilon(a|k; \omega), a \in S(k)\), have an MEV distribution with the CPGF \(G_k(y)\) specified in (4.25) and the structure parameters specified in (4.29), (4.31) and (4.32). Indeed, if \(\omega = 0\) then the NRL model becomes the RL model and if \(\lambda = 0\) then \(\xi^k_r = \xi^k_m\), \(\forall k \in S, m \in S_k(r)\), so the \(G_k\) function in (4.25) becomes

\[
G_k(y) = \sum_{a \in S(k)} y^k_a = \sum_{a \in S(k)} y^\mu_a,
\]

meaning that the choice model at state \(k\) is MNL, and the RCNL model becomes exactly the NRL model. Finally, the maximum likelihood estimation of the RCNL model is a constrained optimization problem and can be expressed as

\[
\max_{\beta, \omega, \lambda \geq 0} LL^{\text{RCNL}}(\beta, \omega, \lambda).
\]

### 4.6.2 Estimation Results

We report the estimation results in Table 4.1 for the four specifications NRL, NRL-LS, RCNL and RCNL-LS. The results are comparable to those previously published using the same data. The \(\beta\) estimates have their expected signs and are highly significant. For both the NRL and RCNL models, the \(\omega\) estimates are negative for travel time and positive for left turns and link constant. All the \(\omega\) estimates for the RCNL models are significantly different from zero. However, for the NRL model, \(\hat{\omega}_{TT}\) is not significantly different from zero when the LS attribute is included in the instantaneous utilities.

We now turn our attention to the \(\lambda\) estimates. The \(\lambda\) estimates are very close to zero for travel time and link size but significantly different from zero for the parameters associated with the
<table>
<thead>
<tr>
<th>Parameters</th>
<th>NRL</th>
<th>NRL-LS</th>
<th>RCNL</th>
<th>RCNL-LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\beta}_{TT}$</td>
<td>-1.854</td>
<td>-2.139</td>
<td>-1.378</td>
<td>-1.567</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.132</td>
<td>0.145</td>
<td>0.080</td>
<td>0.077</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-14.05</td>
<td>-14.75</td>
<td>-17.296</td>
<td>-20.336</td>
</tr>
<tr>
<td>$\hat{\beta}_{LT}$</td>
<td>-0.679</td>
<td>-0.748</td>
<td>-0.517</td>
<td>-0.568</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.043</td>
<td>0.047</td>
<td>0.018</td>
<td>0.019</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-15.79</td>
<td>-15.91</td>
<td>-19.528</td>
<td>-30.105</td>
</tr>
<tr>
<td>$\hat{\beta}_{LC}$</td>
<td>-0.258</td>
<td>-0.224</td>
<td>-0.065</td>
<td>-0.072</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.016</td>
<td>0.015</td>
<td>0.013</td>
<td>0.011</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-16.13</td>
<td>-14.93</td>
<td>-24.656</td>
<td>-6.648</td>
</tr>
<tr>
<td>$\hat{\beta}_{UT}$</td>
<td>-3.340</td>
<td>-3.301</td>
<td>-2.907</td>
<td>-2.964</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.200</td>
<td>0.207</td>
<td>0.094</td>
<td>0.099</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-16.70</td>
<td>-15.95</td>
<td>-30.963</td>
<td>-30.507</td>
</tr>
<tr>
<td>$\hat{\beta}_{LS}$</td>
<td>-0.155</td>
<td>-0.115</td>
<td>-0.115</td>
<td>-0.115</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-0.013</td>
<td>-0.013</td>
<td>0.007</td>
<td>0.007</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-11.92</td>
<td>-11.92</td>
<td>-16.140</td>
<td>-16.140</td>
</tr>
<tr>
<td>$\hat{\omega}_{TT}$</td>
<td>-0.515</td>
<td>-0.341</td>
<td>-0.637</td>
<td>-0.433</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.255</td>
<td>0.288</td>
<td>0.220</td>
<td>0.216</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-2.02</td>
<td>-1.18</td>
<td>-2.898</td>
<td>-2.053</td>
</tr>
<tr>
<td>$\hat{\omega}_{LS}$</td>
<td>0.674</td>
<td>0.581</td>
<td>0.192</td>
<td>0.157</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.003</td>
<td>0.009</td>
<td>0.031</td>
<td>0.025</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>7.25</td>
<td>6.46</td>
<td>6.224</td>
<td>6.275</td>
</tr>
<tr>
<td>$\hat{\omega}_{OL}$</td>
<td>0.086</td>
<td>0.092</td>
<td>0.027</td>
<td>0.021</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.015</td>
<td>0.016</td>
<td>0.003</td>
<td>0.002</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>5.73</td>
<td>5.75</td>
<td>9.846</td>
<td>9.167</td>
</tr>
<tr>
<td>$\hat{\lambda}_{TT}$</td>
<td>-2.63E-04</td>
<td>-2.85E-05</td>
<td>1.84E-08</td>
<td>1.74E-07</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\hat{\lambda}_{LS}$</td>
<td>0.475</td>
<td>0.483</td>
<td>0.437</td>
<td>0.437</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.012</td>
<td>0.012</td>
<td>0.012</td>
<td>0.012</td>
</tr>
<tr>
<td>Rob. $t$-test(0)</td>
<td>41.151</td>
<td>41.230</td>
<td>41.151</td>
<td>41.230</td>
</tr>
</tbody>
</table>

Table 4.1: Estimation results

Note that we do not provide standard errors and $t$-tests for the estimates that are on the bound (close to 0) since the respective gradient component values are not close to zero. The $\lambda$ estimates indicate that only the attribute $OL$ affects the correlation between successor states.

The final LL values are reported in Table 4.1 and we also report the likelihood ratio test in Table 4.2. Similar to Fosgerau et al. (2013a) and Mai et al. (2015c), we observe a significant improvement in final LL values when we include the LS attribute to the instantaneous utilities. The RCNL models have better fit than the NRL models and the best model in term of in-sample fit is the RCNL-LS. Moreover, the final LL function given by the RCNL without the LS is larger than the one given by the NRL model with the LS attribute.

We note that the estimation of the RCNL model requires estimating the NRL model on the integrated network. The real network has 7288 states (i.e. links) and the integrated network has 31373 states, so there are 24084 new states added to the original network. The number of
<table>
<thead>
<tr>
<th>Models</th>
<th>$\chi^2$</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRL &amp; NRL-LS</td>
<td>471.8</td>
<td>1.30e-104</td>
</tr>
<tr>
<td>NRL &amp; RCNL</td>
<td>604.8</td>
<td>9.18e-131</td>
</tr>
<tr>
<td>NRL-LS &amp; RCNL-LS</td>
<td>553.2</td>
<td>1.41e-119</td>
</tr>
<tr>
<td>RCNL &amp; RCNL-LS</td>
<td>420.2</td>
<td>2.21e-93</td>
</tr>
</tbody>
</table>

Table 4.2: Likelihood ratio test results

arcs in the integrated network is 72252, compared to 20202 arcs in the original network. So the estimation of the RCNL model is more expensive than the NRL and RL models. Solving the value functions in the RCNL model needs from 300 to 700 iterations to converge to the fixed point solutions while the NRL needs less than 300 iterations. Furthermore, using the dynamic accuracy and switching approaches, we are able to double the speed of the value iteration method.

### 4.6.3 Prediction Results

In this section we compare the prediction performance of the different models. Similar to Mai et al. (2015c), we use a cross validation approach, i.e., the sample of observations is divided into two sets by drawing observations uniformly: one set contains 80% of the observations is used for estimation, and the other set (20% of the observations) is used as holdout to evaluate the predicted probabilities by applying the estimated model. We generate 20 holdout samples of the same size by reshuffling the real sample and we use the LL loss to evaluate the prediction performance.

For each holdout sample $i$, $0 \leq i \leq 20$, we estimate the parameters $\hat{\beta}_i$ of the corresponding training sample $PS_i$ and this vector of parameters is used to compute the test errors $err_i$

$$err_i = -\frac{1}{|PS_i|} \sum_{\sigma_j \in PS_i} \ln P(\sigma_j, \hat{\beta}_i),$$

where $|PS_i|$ is the size of prediction sample $PS_i$. In order to have unconditional test error values, we compute the average of $err_i$ over samples as follows

$$err_p = \frac{1}{p} \sum_{i=1}^{p} err_i, \quad \forall 1 \leq p \leq 20.$$  \hspace{1cm} (4.33)

The values of $err_p$, $1 \leq p \leq 20$, are plotted in Figure 4.5 and Table 4.3 reports the average of the test error values over 20 samples given by the NRL, NRL-LS, RCNL, RCNL-LS models. The lower test error values the better the model in prediction.
The prediction results show that the models with the LS attribute perform better than those without. The RCNL models have better prediction performances than the NRL models. The RCNL-LS performs the best in fit and prediction among the considered models.

\[
\begin{array}{cccc}
\text{NRL} & \text{NRL-LS} & \text{RCNL} & \text{RCNL-LS} \\
3.34 & 3.21 & 3.17 & 3.07 \\
\end{array}
\]

Table 4.3: Average of test error values over 20 holdout samples

4.7 Conclusion

This paper has considered the RNMEV model, a generalized recursive route choice model, where the choice model at each stage can be any member of the network MEV model. We have introduced a method of integrating correlation structures for simplifying the contraction mapping given by Bellman’s equation of the RNMEV model. This approach allowed us to use the contraction mapping from the NRL model and the methods proposed in Mai et al. (2015c) to quickly estimate the RNMEV model on real data.

We have applied the proposed method to the estimation of the RCNL model, a member of the RNMEV model where the model at each choice stage is a cross-nested logit model. We have showed that the RCNL can exhibit a more general correlation structure, compared to the RL and NRL models. We have provided numerical results using a real data. The RCNL model can be estimated quickly using the new method. Moreover, the parameter estimates are sensible, and the RCNL model has significantly better fit than the NRL model. We have also provided
a cross-validation study suggesting that RCNL and RCNL-LS are better than the NRL and NRL-LS models.

In this paper, we have used an uni-modal network and observations of trips made by car but the model is not restricted to this type of network. In the future research, we plan to apply the RNMEV to different types of networks where the correlation at each choice stage need to be taken into account, e.g., dynamic networks (state is time and location), and multi-modal networks (state is location and mode).

Finally, we note that the method of integrating correlation structures is not restricted to route choice applications. It can be used with the nested fixed point algorithm (Rust, 1987) to estimate dynamic discrete choice models with MEV random terms.
Chapter 5

A Decomposition Method for Estimating Recursive Logit Based Models

We design a decomposition (DeC) method to speed up the estimation of the RL model. More precisely, we show how to reduce the number of linear systems to be solved when estimating the RL model, opening the possibility to estimate complex extensions of this model, for instance mixed RL models. We illustrate the approach on two mixed RL specifications, one using random coefficients and one incorporating error components associated with subnetworks. The models are estimated on a real network, and a cross-validation study is performed.

The application of the DeC method is not restricted to the mixed RL models. It is applied to the regret-based models presented in Chapter 6. The method may also be useful to other route choice applications using the RL model. Moreover, the decomposition idea may be promising to design methods that allow to speed up the estimation of other recursive models, e.g. the NRL (Chapter 3) and the RNMEV (Chapter 4).

The work was presented at the CORS/INFORMS 2015 Joint International Meeting, Montreal, and the 22nd International Symposium on Mathematical Programming (ISMP). The work is also available as CIRRELT technical report (Mai et al., 2015a). The paper has been submitted for possible publication in EURO Journal on Transportation and Logistics.
A Decomposition Method for Estimating Recursive Logit Based Route Choice Models

Tien Mai 1  Fabian Bastin 1  Emma Frejinger 1

Abstract

Fosgerau et al. (2013a) recently proposed the recursive logit (RL) model for route choice problems, that can be consistently estimated and easily used for prediction without any sampling of choice sets. Its estimation however requires solving many large-scale systems of linear equations, which can be computationally costly for real data sets. We design a decomposition (DeC) method in order to reduce the number of linear systems to be solved, opening the possibility to estimate more complex RL based models, such as, mixed RL models.

We illustrate the approach on two mixed RL specifications, one using random coefficients and one incorporating error components associated with subnetwork (Frejinger and Bierlaire, 2007). The models are estimated on a real network with more than 3000 nodes and 7000 links, and a cross-validation study is performed. The results suggest that the DeC method significantly speeds up the estimation of the RL model and allows to estimate the mixed RL models in a reasonable time. The mixed RL models yield sensible parameter estimates and the fit and prediction are significantly better than the RL model.

Keywords: Decomposition method, route choice, mixed recursive logit models, subnetworks, cross-validation

5.1 Introduction

Given a transport network with a number of attributes associated with each link or path, the objective of route choice models is to assign a choice probability to each path in the network. Discrete choice models are typically used, despite two main issues, namely (i) choice sets of paths are unknown to the analyst and the set of all feasible paths for a given origin-destination pair cannot be enumerated in this context, and (ii) path utilities may be correlated, for instance, due to physical overlap in the network. This paper concerns recursive route choice models that can be consistently estimated without sampling of choice sets and allows links and paths to be correlated. The estimation of these models is expensive since it requires solving many systems of linear equations to compute the log-likelihood function and its gradients. We deal with this challenge by proposing a new method that allows to reduce the number of systems to be solved, saving computational time significantly.

1Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
Most of the existing route choice models are based on choice sets of paths that are sampled from the full choice sets. The sampled choice sets can be considered as the actual choice sets (e.g. Ben-Akiva and Bierlaire, 1999a), or can be used to get consistent estimates by correcting the path choice probabilities (Frejinger et al., 2009, Lai and Bierlaire, 2015). Recently, Fosgerau et al. (2013a) propose the recursive logit (RL) model, which considers the set of all feasible paths in the network. The choice probability for each path is computed by means of dynamic programming, the choice of a path being described as a sequence of link decisions. The model can be consistently estimated and is quickly used for prediction without sampling of choice sets. It however retains the well-know independence of irrelevant alternatives (IIA) property which is undesirable in a route choice setting (Mai et al., 2015d). Mai et al. (2015c) propose the nested RL (NRL) model that relaxes the IIA property over paths by assuming that scale parameters are link specific.

The estimation of the RL model requires solving a dynamic programming (DP) problem, which is considerably more time consuming than estimating the MNL model with finite choice sets. The DP problems can become costly to solve, e.g., for very large networks, large number of observations (destinations), or for models that require simulation (mixed logit). In order to address this issue, we propose a decomposition (DeC) method that allows to reduce the number of linear systems to be solved when tackling the DP problem of the RL model. This method speeds up the estimation and opens the possibility to estimate mixed RL models or RL models with nonlinear-in-parameters link utilities.

We apply the DeC to mixed recursive logit models. The mixed logit is attractive model since it relaxes the IIA assumption and is fully flexible, in the sense that it can approximate any random utility model (McFadden and Train, 2000). However, it is rarely used in the context of route choice analysis due its estimation cost. For example, Bekhor et al. (2002) estimate a mixed MNL model based on the Error Component (EC) approach (Bolduc and Ben-Akiva, 1991) using route choice data collected in Boston. Frejinger and Bierlaire (2007) also use the EC approach to model path correlations using subnetwork components. These approaches are however based on generated choice sets of paths. The mixed RL models considered in this paper take advantage of the RL model, as they can be consistently estimated and used for prediction without sampling choice sets, while allowing path and links to be correlated. The main concern is the computational cost of its estimation due to the presence of numerous linear systems. We show how the DeC method can be used for estimating mixed RL models and present estimation results for two mixed RL models based on the Borlänge network in Sweden.

The paper is structured as follows. Section 5.2 reviews the RL model and Section 5.3 introduces the DeC method. The mixed RL model is presented in Section 5.4. The subnetwork error components model is presented in Section 5.5. We provide in Section 5.6 numerical results based on real data, and finally, Section 5.7 concludes.
5.2 Recursive Logit

The RL model proposed in Fosgerau et al. (2013a) is based on the observation that a path choice can be formulated as a sequence of link choices and modeled in a dynamic discrete choice framework. A directed connected network (not assumed acyclic) $G = (\mathcal{A}; \mathcal{V})$ is considered, where $\mathcal{A}$ is the set of links and $\mathcal{V}$ if the set of nodes. For each link $k \in \mathcal{A}$, we denote the set of outgoing links from the sink node of $k$ by $A(k)$. We assume that $D$ destinations are present, and we associate an absorbing state with each destination by extending the network with a set of dummy links $\mathcal{D} = \{d_1, \ldots, d_D\}$, each $d_l$ departing from the sink node of the destination $l$. The set of all links is denoted as $\tilde{\mathcal{A}} = \mathcal{A} \cup \mathcal{D}$. Figure 5.1 shows a simple network with 4 destinations i.e. $d, c, e$ and $f$. Four dummy links $d_1, d_2, d_3$, and $d_4$, without successors, are added to the destinations, respectively.

![Figure 5.1: A small network with multiple destinations](image)

Given a destination $d \in \mathcal{D}$ and two links $k \in \mathcal{A}$, $a \in \mathcal{A} \cup \{d\}$, $a \in A(k)$, the following instantaneous utility

$$ u^n(a|k; \beta) = v^n(a|k; \beta) + \mu(\epsilon(a) - \gamma) \tag{5.1} $$

is associated with action $a \in A(k)$ of individual $n$, where $\beta$ is a vector of parameters to be estimated, $v^n(a|k; \beta)$ is a deterministic utility, and $\gamma$ is Euler’s constant. The deterministic utilities associated with destination $d$ are set to zero, i.e., $v^n(d|k; \beta) = 0$. The random terms $\epsilon(a)$ are assumed to be i.i.d extreme value type I and Euler’s constant is used in order to ensure that the error terms have zero mean. We note that, in the NRL model (Mai et al., 2015c), the scales $\mu$ are assumed to be link specific so the model allows path utilities to be correlated in a fashion similar to the nested logit model (Ben-Akiva, 1973, McFadden, 1978). For notational simplicity, we omit from now an index for individual $n$ but note that the utilities can be individual specific.
As discussed in Fosgerau et al. (2013a), given a link \( k \in A \), the expected maximum utility \( V^d(k; \beta) \), from link \( k \) until the destination \( d \), is given recursively by the logsum

\[
\frac{1}{\mu} V^d(k; \beta) = \ln \left( \sum_{a \in A(k)} e^{\frac{1}{\mu}(v(a|k; \beta) + V^d(a; \beta))} \right) \quad \forall k \in A,
\]

and \( V^d(d; \beta) = 0 \) by assumption. The superscript \( d \) indicates that the value functions are destination specific. The choice probability of a given path \( \sigma = \{k_0, \ldots, k_J\} \) is

\[
P(\sigma; \beta) = e^{-V^d(k_0; \beta)} \prod_{i=1}^{J} e^{v(k_{i+1}|k_i; \beta)}. \tag{5.2}
\]

The vector of parameters \( \beta \) can be estimated by maximizing the log-likelihood (LL) function defined over path observations. The method introduced by (Fosgerau et al., 2013a) to estimate the RL model requires to solve one linear system per destination, at each iteration of the LL maximization, but this is computationally too costly when dealing with complex models such as the mixed logit.

### 5.3 Decomposition Method

In this section we propose a Decomposition (DeC) method that allows to consider only one system of linear equations when computing the value functions associated with all the destinations, thus significantly reducing the computational time of the estimation process.

Given a destination \( d \), we introduce a destination-specific matrix \( M^d \) of size \( (|A| + 1) \times (|A| + 1) \), whose elements are defined for all \( k, a \in A \cup \{d\} \) as

\[
m_{ka} = \begin{cases} 
\delta(a|k)e^{\frac{1}{\mu} v(a|k)} & \text{if } k \in A \\
0 & \text{if } k = d,
\end{cases}
\]

where \( \delta(a|k) = 1 \) if \( a \in A(k) \), and 0 otherwise. We impose that the last column and last row correspond to the dummy link \( d \), so

\[
M^d = \begin{pmatrix}
m_{1,1} & m_{1,2} & \cdots & m_{1,|A|+1} \\
m_{2,1} & m_{2,2} & \cdots & m_{2,|A|+1} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix}.
\]

A vector \( z^d \) of size \( |A| + 1 \) is also defined with elements \( z^d_k = e^{\frac{1}{\mu} V(k)} \) for all states \( k \in A \cup \{d\} \) and \( b^d \) is a vector of size \( |A| + 1 \) with zero values for all states except for dummy \( d \), the corresponding
component being equal to one. Fosgerau et al. (2013a) show that components of \( z \) are solutions to the following linear system

\[
z^d = M^d z^d + b^d.
\]  

(5.3)

We now describe the DeC method based on the set of destinations taken from the observations. We first define a matrix \( M^0 \) of size \((|A| + 1) \times (|A| + 1)\) with entries

\[
m_{ka}^0 = \begin{cases} 
m_{ka} & \text{if } k, a \in A \\
0 & \text{otherwise}
\end{cases}.
\]

Moreover, we define a matrix \( U^d \) of size \((|A| + 1) \times (|A| + 1)\), for each absorbing state \( d \in D \), with entries

\[
u_{ka}^d = \begin{cases} 
\delta(d|k)e^{\frac{1}{\mu} V^d(k)} & \text{if } a = d \\
0 & \text{otherwise}.
\end{cases}
\]

So for each \( d \in D \) we have \( M^d = M^0 + U^d \), or more explicitly

\[
\begin{pmatrix}
m_{1,1} & m_{1,2} & \cdots & m_{1,|A|+1} \\
m_{2,1} & m_{2,2} & \cdots & m_{2,|A|+1} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix} =
\begin{pmatrix}
m_{1,1} & m_{1,2} & \cdots & 0 \\
m_{2,1} & m_{2,2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix}
+ \begin{pmatrix}
0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & m_{2,|A|+1} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix}.
\]

Since \( V^d(d) = 0 \), \( z_{|A|+1}^d = e^{V^d(d)/\mu} = 1 \) and

\[
U^d z^d =
\begin{pmatrix}
0 & 0 & \cdots & m_{1,|A|+1} \\
0 & 0 & \cdots & m_{2,|A|+1} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix}
\begin{pmatrix}
z_1^d \\
z_2^d \\
\vdots \\
z_{|A|+1}^d
\end{pmatrix} =
\begin{pmatrix}
m_{1,|A|+1} \\
m_{2,|A|+1} \\
\vdots \\
0
\end{pmatrix},
\]

we can therefore write (5.3) as

\[
z^d = M^0 z^d + U^d z^d + b^d = M^0 z^d + t^d + b^d,
\]

where \( t^d \) is the last column of the matrix \( U^d \). Let \( Z \) be a matrix of size \((|A| + 1) \times |D|\) whose columns are the vectors \( z^d \), for all \( d \in D \)

\[
Z = [z^{d_1}, \ldots, z^{d_{|D|}}],
\]
and $B$ be a matrix of size $(|A| + 1) \times |D|$ whose columns are the vectors $t^d + b^d$, $\forall d \in D$. We obtain a new system of linear equations

$$Z = M^0 Z + B$$

equivalent to

$$(I - M^0)Z = B. \tag{5.4}$$

The system has a solution if $I - M^0$ is invertible. The conditions to ensure that $I - M^0$ is invertible are discussed in Fosgerau et al. (2013a), and they can be extended to $I - M^0$. In essence, the existence of a solution to the above system depends on the magnitude of the link instantaneous utilities. It is easy to find a feasible solution by using large enough magnitude of the model parameters.

For a given dummy $d_t \in D$ and a link $k \in A$, the corresponding value function is

$$V^{d_t}(k) = \mu \ln(z_{kd_t}),$$

where $z_{kd_t}$ is the element of matrix $Z$ corresponding to column $d_t$ and row $k$. Hence, the associated LL function is

$$LL(\beta) = \frac{1}{N} \sum_{n=1}^{N} (v(\sigma_n, \beta) - \ln z_{o_n,d_n}(\beta)),$$

where $v(\sigma_n, \beta) = \sum_{i=0}^{l_n-1} v(k_{i+1}^n | k_i^n)$ is the sum of the deterministic link utilities of path $\sigma_n$ and $o_n, d_n$ are the origin and dummy link of path $\sigma_n$, respectively. The gradient and Hessian of the LL function require the derivatives of the value functions. These can be obtained by deriving the Jacobian of vector $Z$ with respect to a parameter $\beta_i$ as

$$\frac{\partial Z}{\partial \beta_i} = (I - M^0)^{-1} \frac{\partial M^0}{\partial \beta_i} Z + (I - M^0)^{-1} \frac{\partial B}{\partial \beta_i},$$

and the Hessian with respect to two parameters $\beta_i, \beta_j$ is

$$\frac{\partial^2 Z}{\partial \beta_i \partial \beta_j} = (I - M^0)^{-1} \left( \frac{\partial^2 M^0}{\partial \beta_i \partial \beta_j} Z + \frac{\partial M^0}{\partial \beta_i} \frac{\partial Z}{\partial \beta_j} + \frac{\partial M^0}{\partial \beta_j} \frac{\partial Z}{\partial \beta_i} + \frac{\partial^2 B}{\partial \beta_i \partial \beta_j} \right).$$

Hence, we only need to solve the system of linear equations (5.4) once, and vector $Z$ contains all the value functions for all destinations.

Finally, it is important to note that the RL model and its extensions may be useful for traffic simulation. In this context, one needs to compute the next-link choice probabilities $P(a|k)$, $\forall k \in A, a \in A(k)$, and link flows in the network. The next-link choice probabilities given by matrix $P$ are computed using the value functions, and the link flows can be obtained by
inverting matrix $I - P^T$, where $I$ is the identity matrix. Baillon and Cominetti (2008) show that $I - P^T$ is invertible. The DeC method allows to compute matrix $P$ quickly, hence it is useful for prediction and simulation, in addition to the maximum likelihood estimation.

### 5.4 Mixed Recursive Logit

The choice probability of a path given by the RL model can be computed using (5.2). In the mixed multinomial logit framework, the values $v(k_{i+1}|k_i; \beta)$ themselves contain random elements. Following Train (2003), we assume that $\beta$ is derived from a random vector $\omega$ and a parameters vector $\theta$, i.e. $\beta = \beta(\theta, \omega)$. For example, if $\beta$ is a $K$-dimensional normally distributed random vector, whose components are mutually independent, we may chose $\omega = (\omega_1, \ldots, \omega_K)$ with $\omega_i \sim N(0,1)$, $i = 1, \ldots, K$ and let $\theta$ specify the means and the standard derivations of the components of $\beta$. The unconditional choice probability is obtained by taking the expectation over the random coefficients

$$P(\sigma; \theta) = \mathbb{E}_\omega[P(\sigma; \beta)] = \int P(\sigma; \theta, \omega)f(\omega)d\omega,$$

where the expectation is taken with respect to $\omega$ and $f(\cdot)$ is its density function. The vector $\theta$ can be estimated by maximizing the LL function defined over the set of path observations $n = 1, \ldots, N$

$$LL(\theta) = \frac{1}{N} \sum_{n=1}^{N} \ln P(\sigma_n; \theta).$$

This involves the computation of $P(\sigma_n; \theta)$ for each observation, and therefore, by (5.5), one multidimensional integral per individual. An analytical expression of (5.5) can usually not be obtained, it therefore has to be numerically approximated, either by quadrature methods, or by simulation.

Assuming that the integral dimension is $K$, (5.5) is approximated as

$$\tilde{P}(\sigma_n; \theta) = \sum_{i=1}^{R_n} w_{n,i} P(\sigma_n; \theta, \omega_{n,i}),$$

where $\omega_{n,i}$ are the integration nodes and $w_{n,i}$ are the integrations weights. The nodes and weights can be deterministically produced or, for Monte-Carlo methods, randomly drawn from the distribution of $\omega$, and (5.6) becomes

$$\tilde{P}(\sigma_n; \theta) = \frac{1}{R_n} \sum_{r=1}^{R_n} P(\sigma_n; \theta, \omega_{n,i}).$$
The Monte Carlo approach better scales with the integral dimension, and under mild conditions, the estimators derived from the simulated log-likelihood converge almost surely towards the true maximum likelihood estimators as the number of draws $R^m$ tends to infinity (Bastin et al., 2006b). Randomized quasi-Monte Carlo methods have also been considered for mixed logit models, while the improvement is sometimes limited (Munger et al., 2012).

A major burden is however that different integration nodes produce different linear systems of the form (5.4), that have to be solved, inducing a significant numerical cost increase. We can limit it by requiring the integration nodes to be the same among the observations, i.e.

\[ w_{n,i} \equiv w_i, \quad \omega_{n,i} \equiv \omega_i, \quad n = 1, \ldots, N. \]

The integration error still converges to zero as the number of integration is growing to infinity, the convergence being almost sure in the case of Monte Carlo methods. In the latter case, the simulated choice probabilities are no longer independent between the observations, resulting in an additional simulation bias that nevertheless goes to zero asymptotically with the number of draws, and in practice, it can often be neglected.

In order to illustrate this point, we compare the estimates obtained by the two approaches using the small network in Figure 5.1. The network is composed of 15 nodes and 28 arcs. There are two origin nodes $a, b$ and four destination nodes which are $c, d, e, f$. Note that each origin connects to all destinations, so there are eight OD pairs in total. Travel time is the only attribute considered in our example and they are generated uniformly in interval $[0, 1]$. We assume that each link $a$ is associated with a travel time $TT(a)$ and we consider the following instantaneous utility with respect to link $a$ given link $k$

\[ u(a|k) = \beta_1 TT(a) + \beta_2 + (\epsilon(a) - \gamma), \quad (5.8) \]

where $\epsilon(a)$ follows an extreme value type I, $\gamma$ is Euler’s constant, $\beta_1$ and $\beta_2$ are normally distributed. We assume that $\beta_1 = N(\theta_{TT}, \sigma_{TT}^2)$ and $\beta_2 = N(0, \sigma^2)$. The path observations are simulated using parameters $\{\theta_{TT}^0, \sigma_{TT}^0, \sigma^0\} = \{-11, 0.1, 0.1\}$. 5 paths are simulated for each OD pair, so the sample contains 40 path observations in total. We note that the number of observations is kept small because we do not aim at recovering the true parameters and even with small number of observations, the LL function is well defined.
We estimate the mixed RL model with CRN and IRN approaches. All the reported estimated parameters and final LL values are based on 100 independent simulations. For each estimation with CRN or IRN we use 50, 200 and 1000 Monte Carlo draws. The means and standard derivations over 100 simulations are reported in Table 5.1. We also compute a t-test for the null hypothesis that the mean of the estimated parameters given by the CRN is not significant different from its respective value given by the IRN approach. The results show that the estimated parameters as well as the final LL given by the CRN method are not significantly different with the ones given by the IRN method at the 0.05 significance level. As expected, the standard deviations are larger for the lower numbers of draws. The parameter estimates of CRN and IRN can be different (e.g. 0.089 versus 0.461) without being significantly different. When the number of draws is large enough (e.g. 1000) the LL and parameter estimates have similar values and small standard deviations. In summary, for this example, the CRN method is an alternative to the standard IRN.

<table>
<thead>
<tr>
<th>Number of draws</th>
<th>CRN</th>
<th>IRN</th>
<th>CRN</th>
<th>IRN</th>
<th>CRN</th>
<th>IRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL($\beta$)</td>
<td>0.699</td>
<td>0.698</td>
<td>0.700</td>
<td>0.699</td>
<td>0.701</td>
<td>0.701</td>
</tr>
<tr>
<td>std. dev.</td>
<td>0.002</td>
<td>0.003</td>
<td>0.001</td>
<td>0.001</td>
<td>0.0005</td>
<td>0.0008</td>
</tr>
<tr>
<td>t-test</td>
<td>0.137</td>
<td>-</td>
<td>0.433</td>
<td>-</td>
<td>0.137</td>
<td>-</td>
</tr>
<tr>
<td>std. dev.</td>
<td>0.554</td>
<td>0.597</td>
<td>0.289</td>
<td>0.440</td>
<td>0.136</td>
<td>0.400</td>
</tr>
<tr>
<td>t-test</td>
<td>-1.044</td>
<td>-</td>
<td>-0.273</td>
<td>-</td>
<td>-0.226</td>
<td>-</td>
</tr>
<tr>
<td>$\sigma_{TT}$</td>
<td>0.284</td>
<td>0.971</td>
<td>0.089</td>
<td>0.461</td>
<td>0.107</td>
<td>0.193</td>
</tr>
<tr>
<td>std. dev.</td>
<td>0.161</td>
<td>0.683</td>
<td>0.077</td>
<td>0.437</td>
<td>0.069</td>
<td>0.184</td>
</tr>
<tr>
<td>t-test</td>
<td>-1.007</td>
<td>-</td>
<td>-0.849</td>
<td>-</td>
<td>-0.465</td>
<td>-</td>
</tr>
<tr>
<td>$\bar{\sigma}$</td>
<td>0.670</td>
<td>0.479</td>
<td>0.553</td>
<td>0.487</td>
<td>0.501</td>
<td>0.417</td>
</tr>
<tr>
<td>std. dev.</td>
<td>0.121</td>
<td>0.110</td>
<td>0.069</td>
<td>0.113</td>
<td>0.035</td>
<td>0.165</td>
</tr>
<tr>
<td>t-test</td>
<td>1.730</td>
<td>-</td>
<td>0.585</td>
<td>-</td>
<td>0.514</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.1: Estimation results with IRN and CRN

Finally, we note that, thanks to the DeC method, the computational time for estimating the model using the CRN is from 10 to 50 times less than when estimating the model using the IRN. We also observe that, for the CRN, the optimization algorithm often converges faster in terms of number of iterations, compared to the conventional IRN.

5.5 Modeling Correlations with Subnetwork Components

In this section we present a route choice modeling approach that is convenient to use with the mixed RL model. It was proposed by Frejinger and Bierlaire (2007), and it as called the subnetwork approach. A subnetwork is a set of components where each component is defined as a sequence of links. A component can, for instance, corresponds to a main road in the transport network. The approach is based on the assumption that paths may be correlated if they share a subnetwork component. To capture this correlation, Frejinger and Bierlaire (2007) propose an
Error Component (EC) model (see for instance Bekhor et al., 2002). Inspired by this approach, in the following, we propose an EC model for the mixed RL model that allows to capture the correlation between paths and links that share subnetwork components in the network.

Assuming that the network is composed of $Q$ subnetwork components. We add error component factors to the instantaneous utilities as

$$u(a|k; \beta, \sigma) = v(a|k; \beta) + F(a)^T T(\sigma) \zeta + \mu(\epsilon(a) - \gamma), \ \forall k \in \mathcal{A}, a \in A(k),$$

where $F(a)$ is a vector of dimension $Q$ and each element $F(a)_q$ associates link $a$ in overlaps with subnetwork component $q$ (i.e. if link $a$ is a part subnetwork component $q$, $F(a)_q = \sqrt{l(a)}$ and $F(a)_q = 0$ otherwise, where $l(a)$ is the length of link $a$), $T(\sigma) = \text{diag}(\sigma_1, \ldots, \sigma_Q)$ where $\sigma$ is a vector of covariance parameters to be estimated, and $\zeta$ is a vector of size $Q$ where each element is a $N(0, 1)$ random variable.

### 5.6 Numerical Results

In this section we use the same data and network used in Frejinger and Bierlaire (2007), Fosgerau et al. (2013a) and Mai et al. (2015c) to apply the DeC method and provide estimation and prediction results for the mixed RL models with random parameters and error components. The network consists of of 3077 nodes and 7459 links, and with static and deterministic travel times. The sample consists of 1832 trips corresponding to simple paths with a minimum of five links. There are 466 destinations, 1420 different origin-destination (OD) pairs and more than 37,000 link choices in this sample. For the sake of comparison, we also estimate the RL models (Fosgerau et al., 2013a).

As in Frejinger and Bierlaire (2007), the subnetwork components of the Borlänge network are defined based on the main roads for traversing the city center. Two of the Swedish national roads traverse Borlänge. The subnetwork is composed of these national roads (referred to as R.50 and R.70) and there are two subnetwork components for each national road (north and south directions), leading to a total of four subnetwork components, denoted R.70N, R.70S, R.50S and R.50N. In addition, there is one component for the road segment in the city center which is denoted RC. Figure 5.2 shows the Borlänge network and 5 subnetwork components. Reader can consult Frejinger and Bierlaire (2007) for more details.

### 5.6.1 Specification of Models

The same four attributes as in Fosgerau et al. (2013a) are used in the instantaneous utilities: link travel time $TT(a)$, a left turn dummy $LT(a|k)$ that equals to one if the turn angle from
link $k$ to $a$ is larger than 40 degrees and less than 177 degrees, a u-turn dummy $UT(a|k)$ that equals to one if the turn angle is larger than 177, and a link constant $LC(a)$ set to 1 for all links in our experiments. Fosgerau et al. (2013a) also propose the link size (LS) attribute for overlapping paths. This attribute is however origin-destination specific and is not compatible with the DeC model, therefore we do not use this attribute for the mixed RL models.

We specify the instantaneous utilities for different models for link $a$ given link $k$, $a \in A(k)$ as

- **RL**
  \[
  u^{RL}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) \\
  + \beta_{UT}UT(a|k) + (\epsilon(a) - \gamma),
  \]

- **RL with the LS attribute (RL-LS)**
  \[
  u^{RL-LS}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) \\
  + \beta_{UT}UT(a|k) + \beta_{LS}LS(a) + (\epsilon(a) - \gamma),
  \]

- **mixed RL with random parameters (MRL-RP)**
  \[
  u^{MRL}_{RP}(a|k; \beta) = \beta_{TT}^*TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) \\
  + \beta_{UT}UT(a|k) + (\epsilon(a) - \gamma),
  \]
• mixed RL with EC (MRL-EC)

\[ u_{MRL}^{EC}(a|k; \beta, \sigma) = \beta_{TT}^{TT}(a) + \beta_{LT}^{LT}(a|k) + \beta_{LC}^{LC}(a) \]
\[ + \beta_{UT}^{UT}(a|k) + F(a)^T \sigma \zeta + (\epsilon(a) - \gamma), \]

• mixed RL with EC and random parameters (MRL-REC)

\[ u_{MRL}^{REC}(a|k; \beta, \sigma) = \beta_{TT}^{*TT}(a) + \beta_{LT}^{LT}(a|k) + \beta_{LC}^{LC}(a) \]
\[ + \beta_{UT}^{UT}(a|k) + F(a)^T \sigma \zeta + (\epsilon(a) - \gamma), \]

where \( \epsilon(a) \) are i.i.d standard extreme value type I, \( \beta_{TT}^{*TT} \) is specified to be normal distribution \( \beta_{TT}^{*TT} \sim N(\mu_{TT}, \sigma_{TT}^2) \), \( \zeta \) is a vector of size 5 where each element is a \( N(0,1) \) random variable, and \( F(a) \) is a vector of size 5 where each element \( F(a)_q \) with respect to link \( a \) and subnetwork component \( q \) is defined as

\[ F(a)_q = \begin{cases} 0 & \text{if } a \notin q \\ \sqrt{TT(a)} & \text{otherwise.} \end{cases} \]

The vector of the covariance parameters is \( \sigma = \{\sigma_{R50N}, \sigma_{R50S}, \sigma_{R70N}, \sigma_{R70S}, \sigma_{RC}\} \).

5.6.2 Estimation Results

For the estimation of the mixed RL model we adopt the nested fixed point algorithm designed by Rust (1987). This algorithm combines an outer iterative non-linear optimization algorithm for searching over the parameter space with an inner algorithm for solving the value functions. Note that the value functions are solved for each MC draw using the DeC method and we use the basic trust region method (for instance Nocedal and Wright, 2006) with BFGS (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970) as the non-linear optimization algorithm.

We estimate the mixed RL models (MRL-RP, MRL-EC, MRL-REC) and report the results based on 10 independent simulations and for each simulation we use \( R = 500 \) Monte Carlo random draws. The parameter estimates are reported in the appendix, Tables B.1, B.2, and B.3. The \( \beta \) estimates are comparable to those previously published using the same data. The parameter estimates have their expected signs and are all highly significantly different from zero. Over 10 simulations, the \( \beta \) estimates are closer in values, compared to the \( \sigma \) estimates. We note that Frejinger and Bierlaire (2007) estimate path-based models using the same data and the subnetwork approach, and report that all the \( \sigma \) estimates are significantly different from zero except \( \hat{\sigma}_{R50S} \).

We plot the final LL values over 10 repetitions in Figure 5.3. The averaged final LL values over 10 MC repetitions are reported in Table 5.2. For the sake of comparison we also include the
RL models (RL and RL-LS). We only report the final LLs given by these models and reader can consult Fosgerau et al. (2013a) for more details of the estimation results. The mixed RL models with EC performs better than the RL models in fit (the likelihood ratio test results are reported in Table 5.3 based on the averaged final LL values in Table 5.2).

Before presenting the prediction results, we make some remarks on computational time. The code is implemented in MATLAB 2013a (available upon request) and we use an Intel(R) machine, CoreTM i5-3210M CPU 2.50GHz. We use the trust region algorithm with BFGS to estimate all the models. The estimation of the mixed RL models is much more costly, compared to the RL models. We need from 3 to 5 days to estimate the mixed RL models, while it takes half day for estimating the RL models.

<table>
<thead>
<tr>
<th>Models</th>
<th>$\chi^2$</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL &amp; MRL-RP</td>
<td>148.42</td>
<td>3.84e-34</td>
</tr>
<tr>
<td>RL &amp; MRL-EC</td>
<td>667.22</td>
<td>6.00e-142</td>
</tr>
<tr>
<td>MRL-RP &amp; MRL-REC</td>
<td>616.16</td>
<td>6.52e-131</td>
</tr>
<tr>
<td>MRL-EC &amp; MRL-REC</td>
<td>97.36</td>
<td>5.78e-23</td>
</tr>
</tbody>
</table>

Table 5.3: Likelihood ratio test results
5.6.3 Prediction Results

In this section we use a cross validation approach to compare the prediction performance of the different models considered above, i.e., the RL and mixed RL models. The real sample (1832 observations) is divided into two sets by drawing observations uniformly: one set of 80% of observations is used for estimation, and one set of the remaining 20% observations is used as holdout to evaluate the predicted probabilities. We generate 20 different holdout samples of the same size by reshuffling the real sample. The LL loss values then are used to evaluate the prediction performance.

For each holdout sample $i$, $0 \leq i \leq 20$, we estimate the parameters $\hat{\theta}_i$ based on the respective estimation sample, and we compute the test errors $err_i$ using the holdout sample and these parameter estimates as

$$err_i = -\frac{1}{|PS_i|} \sum_{\sigma_j \in PS_i} \ln \hat{P}(\sigma_j, \hat{\theta}_i),$$

where $PS_i$ is the set of observations corresponding to holdout sample $i$, and $|PS_i|$ is the size of $PS_i$. Indeed, $err_i$ depends on $PS_i$. Similarly to Mai et al. (2015c), we compute the average of $err_i$ values over samples in order to have unconditional test error values

$$\overline{err}_p = \frac{1}{p} \sum_{i=1}^{p} err_i \quad \forall 1 \leq p \leq 20.$$ (5.9)

We plot the values of $\overline{err}_p$, $1 \leq p \leq 20$ in Figure 5.4 (lower the value, the better is the performance) and in Table 5.4, we report the average of the test error values given by five models over 20 samples. Indeed, the value of $\overline{err}_p$ becomes stable when $p$ increases for each model. The mixed RL models are better than the RL model in term of prediction. The MRL-REC has a significant better fit and also a better prediction performance, compared to the other models.

<table>
<thead>
<tr>
<th>RL</th>
<th>RL-LS</th>
<th>MRL-RP</th>
<th>MRL-EC</th>
<th>MRL-REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.39</td>
<td>3.34</td>
<td>3.36</td>
<td>3.22</td>
<td>3.19</td>
</tr>
</tbody>
</table>

Table 5.4: Average of test error values over 40 holdout samples

5.6.4 Evaluation of Computational Time for the Decomposition Method

In this section, we provide a comparison of computational time to show how the DeC method speeds up the estimation of the RL model. We compare the performance of the DeC method with the original approach based on the same data used in the previous section.
For the sake of comparison we estimate the path-based logit models proposed by Frejinger et al. (2009) based on the same data set. The models require sampling of choice sets and we draw 50 samples for each OD pair observation. The computational times associated with evaluating the LL and gradients are reported for two models, one with and one without the Extended Path Size attribute (Frejinger et al., 2009). We denote them PL and PSL, respectively. Recall that all the models are estimated with 1832 path observations (466 destinations).

The code is also implemented in MATLAB and it has not been parallelized. The computational times are reported in Table 5.5. For each “LL and gradients” evaluation, the DeC method is about 30 times faster, compared to the original method. The computations associated with the path-based logit models are also fast because the LL can be directly evaluated without computing the value functions. However, the total computational time for the estimation of the path-based models is high because choice sets need to be sampled before estimating. Finally, we note that the RL model with LS attribute is costly to estimate, compared to the model without.

<table>
<thead>
<tr>
<th>Models</th>
<th>DeC method</th>
<th>Original method</th>
<th>Path-based models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>RL</td>
<td>PL</td>
</tr>
<tr>
<td>LL &amp; gradients</td>
<td>10</td>
<td>284</td>
<td>1274</td>
</tr>
<tr>
<td>Estimation</td>
<td>230</td>
<td>18279</td>
<td>29638</td>
</tr>
</tbody>
</table>

Table 5.5: Comparison of computational times (in seconds)
5.7 Conclusion

In this paper we proposed a decomposition (DeC) method to speed up the estimation of the RL models. We have shown that the DeC approach can be used to estimate the mixed RL model in reasonable time. We provided estimation and prediction results using a real data set. We estimated the mixed RL models based on two different approaches which use random parameters and EC using subnetwork components. The results showed that the MRL-REC model has significantly better fit and better prediction than the RL models (with and without LS attribute) and other mixed RL models.

The DeC method significantly speeds up the estimation of the RL model. Our next steps will be dedicated to applying and testing the DeC method to other complex extensions of the RL model, e.g., the NRL model (Mai et al., 2015c), the RL model for stochastic time-dependent networks and regret-based RL models. We are also interested in applying the mixed RL to other networks with panel data and different EC models.
Chapter 6

Comparing Regret Minimization and Utility Maximization

In this chapter we focus on the comparison of estimation and prediction results between the random utility maximization (RUM) and random regret minimization (RRM) decision rules using recursive route choice models. This chapter contributes to the literature on route choice and RRM models by showing how the RL model can be formulated within the RRM framework. We extend an existing RRM model by allowing to capture the impact of non-chosen alternatives in a flexible way. Moreover, in order to have a fair comparison between the RRM and RUM, we propose nonlinear-in-parameters RUM models which have similar forms as the proposed RRM-based models. We present how to estimate these models by maximum likelihood, and we provide comparisons of estimation results as well as in-sample and out-of-sample fit using real data.

The estimation of RRM-based models is expensive since each link needs to be compared with every alternative link in terms of every attribute. We show that the DeC method proposed in Chapter 5 can be used to significantly reduce the computational cost. It is also important to note that the RRM framework is too costly to use with the NRL and RNMEV models (Chapters 3, 4), as the DeC is not compatible with these models.

The work has been presented at the 4th International Choice Modeling Conference, Austin, Texas, U.S.A., May 2015. A preliminary version of the paper is available as CIRRELT technical report (Mai et al., 2015b). The paper is currently under review for possible publication in Journal of Choice Modeling (a first revision has been submitted).
Comparing Regret Minimization and Utility Maximization for Route Choice Using the Recursive Logit Model

Tien Mai¹  Fabian Bastin ¹  Emma Frejinger ¹

Abstract

This paper focuses on the comparison of estimation and prediction results between the random utility maximization (RUM) and random regret minimization (RRM) frameworks for route choice recursive logit (RL) model (Fosgerau et al., 2013a). The RL model is originally based on RUM and in this paper we propose different versions of the RL model based on RRM by adapting and extending the model proposed by Chorus (2014). In order to make a fair comparison between RRM and RUM, we also propose nonlinear-in-parameters RUM models which are similar to the RRM models. We report estimation results and a cross-validation study for a real network with more than 3000 nodes and 7000 links. In our experiments, the cross-validation results show that one of the proposed extended versions of the RRM model has the best out-of-sample fit. While this observation favors the RRM framework, we note that the RRM models are more time consuming to estimate and the parameter estimates are less straightforward to interpret than the RUM ones.

Keywords: Route choice modeling; Recursive logit; Random utility maximization; Random regret minimization; Maximum likelihood estimation; Cross-validation.

6.1 Introduction

Discrete choice models are often used for analyzing path choices in real networks based on revealed preference (RP) data. Following the discussion in Fosgerau et al. (2013a) the route choice models in the literature can be grouped into three approaches. First, the classical approach corresponds to path logit (PL) models where choice sets of paths are generated and treated as the actual choice sets. Second, the sampling approach (Frejinger et al., 2009, Lai and Bierlaire, 2015) is based on the idea that the choice set can be sampled and the estimation can be consistent if sampling corrections are added to the choice probabilities. Recently, Fosgerau et al. (2013a) proposed a third approach, called the recursive logit (RL) model which can be consistently estimated based on RP data and used for prediction without sampling any choice

¹Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
sets of paths. Another extension of the RL model, the nested RL (NRL) model, has been proposed by Mai et al. (2015c) that allows to relax the independence of irrelevant alternatives (IIA) property. These models are based on the Random Utility Maximization (RUM) framework.

Recently, Chorus (2010, 2012, 2014) proposed the Random Regret Minimization (RRM) framework for discrete choice modeling. It is based on a regret minimization decision rule postulating that when decision makers choose between alternatives, they try to avoid the situation where a non-chosen alternative outperforms a chosen one in terms of attributes. We base this paper on the Generalized Random Regret Minimization (GRRM) model proposed by Chorus (2014), and we compare the estimation and prediction results of the RL model using RRM and RUM decision rules.

It is important to note that the regret decision theory is originally developed by Bell (1982, 1983), Loomes and Sugden (1982, 1987), and subsequently improved by several studies. The main idea is that a decision-maker can anticipate the risk to experience regret about some decisions when more information is obtained, typically when some uncertainty is resolved. In our context, the link attributes are assumed to be deterministic and there is no additional information gathered when traveling. We however incorporate the regret theory to our route choice applications and develop models based on the framework proposed by Chorus (2010) in order to take advantages from nonlinear cost functions, and account for attribute difference terms.

Prato (2014) analyses path-based route choice model estimation results of path-based models using the regret model proposed by Chorus (2010). He focuses on the two well-known challenges associated with route choice modeling, namely, choice set generation and correlation. He finds that RRM performs well on real data, but in an experimental setting, he finds that the parameter estimates of RRM models have the wrong signs when “irrelevant alternatives” are included in the choice sets. The RL model is based on the universal choice set of all paths connecting an origin-destination pair and does not need choice set generation. We investigate whether the RL model presents similar issues to path-based models and we analyze the out-of-sample fit.

This paper makes a number of contributions. We propose two specifications for RL models with random regret. The first model (called Extended Random Regret Minimization - ERRM) extends the GRRM model by adding terms associated with the attributes of the non-chosen alternatives to the attribute differences in the regrets. The second model, called Averaged Random Regret Minimization (ARRM) model, modifies the first one by averaging the regrets over the alternatives. We prove that under some constraints on the parameters, the regret given by the ARRM model has a linear-in-parameters form and the corresponding choice probabilities are equivalent to those given by the RUM model proposed in Fosgerau et al. (2013a). This model therefore generalizes the RUM-based RL model.
Furthermore, we propose three nonlinear-in-parameters RUM models which have similar forms as the RRM models. The objective is to provide a fair comparison between the RRM and RUM. Finally, we report estimation and cross-validation results for a real network with over 3000 nodes and 7000 links. The estimation code for the RRM and RUM RL models is implemented in MATLAB and is freely available upon request.

The paper is structured as follows. In Section 6.2 we review the RUM and RRM models. In Section 6.3 we propose the RL model under RRM, and Section 6.4 presents two different formulations for link regrets. Nonlinear-in-parameters RUM models are presented in Section 6.5 and in Section 6.6 we provide an illustrative example. In Section 6.7 we discuss maximum likelihood estimation. Model specifications as well as estimation and cross-validation results are presented in Section 6.8, and finally, Section 6.9 concludes.

6.2 Random utility maximization and random regret minimization models

The RL model is developed based on the dynamic discrete choice framework proposed by Rust (1987). In this context, the path choice problem is formulated as a sequence of link choices, and at each stage the travelers chooses a next link by observing the costs of the outgoing links and the expected costs from the current state to the destination. The link choice probabilities then can be computed by defining link costs and a decision rule at each choice stage. In Fosgerau et al. (2013a), the RUM decision rule is used at each stage, i.e., they assume that the traveler aims at maximizing the sum of the random utility of outgoing links (instantaneous utility) and the expected maximum utility from the sink node of the links to the destination (value function). The random terms of the instantaneous utilities are assumed to be independently and identically distributed (i.i.d.) extreme value type I and the RL model is equivalent to a logit model over an infinite number of path alternatives. Within the RRM decision rule, at each stage we can define the link costs as the regrets associated with all the alternative links, and the expected costs as the expected minimum regrets. This paper focuses on comparing the RUM and the RRM decision rules for the choice at each stage of the RL model. We therefore start by presenting the principles of the RUM and RRM frameworks.

In the context of the RUM discrete choice models, we assume that an individual $n$ associates a utility $U_{ni}$ with an alternative $i$ within a choice set $C_n$. The utility consists of two additive parts $U_{ni} = V_{ni} + \epsilon_{ni}$: a deterministic $V_{ni}$ part, observed by the modeler, and a random part $\epsilon_{ni}$. Typically, $V_{ni}$ a linear-in-parameters function of attributes, i.e., $V_{ni} = \beta^T x_{ni}$, where $\beta$ is a vector of parameters to be estimated and $x_{ni}$ is a vector of attributes with respect to individual
n and alternative i. A decision maker chooses the alternative that maximizes his/her utility

\[ i^* = \arg\max_{i \in C_n} \{V_{ni} + \epsilon_{ni}\}. \]

The well-known multinomial logit (MNL) model is based on the assumption that the random terms \( \epsilon_{ni} \) are i.i.d. extreme value type I, and the probability of choosing an alternative \( i \) is

\[ P_n(i) = \frac{e^{V_{ni}}}{\sum_{j \in C_n} e^{V_{nj}}}. \]

The MNL model however retains the IIA property, so other models may be preferred in order to capture the correlations between random terms e.g. the nested logit model (Ben-Akiva, 1973), cross-nested logit model (Vovsha and Bekhor, 1998) or network multivariate extreme value model (Daly and Bierlaire, 2006). Mai et al. (2015c) propose a nested version of the RL model that relaxes the IIA property. In route choice applications, MNL models with a path size attribute (Ben-Akiva and Bierlaire, 1999a) or a commonality factor (Cascetta et al., 1996) are often used.

The RRM models are based on the assumption that when decision makers choose between alternatives, they try to avoid the situation where a non-chosen alternative outperforms a chosen one in terms of one or more attributes. This translates into a regret function for a considered alternative by including all attributes of all competing alternatives. The random regret \( RR_{ni} \) can be written as the sum of a deterministic part \( R_{ni} \) and a random error term \( \epsilon_{ni} \) (Chorus, 2012),

\[ RR_{ni} = R_{ni} + \epsilon_{ni} = \sum_{j \neq i, j \in C_n} \sum_{t} \ln \left( 1 + e^{\beta_t(x_{nj}(t) - x_{ni}(t))} \right) + \epsilon_{ni} \quad (6.1) \]

where \( t \) is an attribute. The regret \( R_{ni} \) is thus computed based on two sums, the first is over all other alternatives in the choice set and the second over all the attributes. Contrary to the RUM models, a decision maker aims to minimize the random regret

\[ i^* = \arg\min_{i \in C_n} \{R_{ni} + \epsilon_{ni}\} = \arg\max_{i \in C_n} \{-R_{ni} - \epsilon_{ni}\}. \quad (6.2) \]

Under the assumption that the random terms \(-\epsilon_{ni}\) are i.i.d extreme value type I, the choice probability is given by the MNL model

\[ P_n(i) = \frac{e^{-R_{ni}}}{\sum_j e^{-R_{nj}}}. \]

It is important to note that even though this is the logit model, the IIA property does not hold since the regrets are not alternative specific. Chorus (2014) presents the Generalized Random Regret Minimization (GRRM) model, where the random regret can be expressed as

\[ GRR_{ni} = \sum_{j \neq i, j \in C_n} \sum_{t} \ln \left( \lambda_t + e^{\beta_t(x_{nj}(t) - x_{ni}(t))} \right) + \epsilon_{ni}. \quad (6.3) \]
We note that $GRR_{ni} = RR_{ni}$ if $\lambda_t = 1$, $\forall t$. Moreover, as pointed out in Chorus (2014), if $\lambda_t = 0$ $\forall t$ the resulting regret becomes linear-in-parameters

$$GRR_{ni} = \sum_{j \neq i \in C_n} \sum_t \beta_t(x_{nj}(t) - x_{ni}(t)) - \sum_{j \in C_n} \beta^T x_{nj} - |C_n|\beta^T x_{ni},$$

where $|\cdot|$ is the cardinality operator. The term $\sum_{j \in C_n} \beta^T x_{nj}$ is the same whatever the considered alternative $i$, and does not affect the choice given by (6.2). The regret has a linear-in-parameter formulation but it is different from the RUM model because of $|C_n|$.

A disadvantage of the RRM or GRRM model, highlighted in Chorus (2012), is that the running time for computing the choice probabilities increases exponentially with the size of the choice. Indeed, every alternative is compared with every other in terms of each attribute. This can hence be a problem for path-based route choice applications which are characterized by large choice sets. While the RL model is based on the universal choice set (of infinite size), the choice set at each choice stage is small (outgoing links at a node). This is therefore not an issue for the RL model.

Finally, we note that the random regrets $GRR_{ni}$ and $RR_{ni}$ are undefined when the choice set $C_n$ is singleton. This is an issue for the RL model if the transport network contains only one outgoing link for some nodes. As we explain in the following, we deal with this issue by summing over all alternatives.

### 6.3 Random regret recursive logit models

In the RUM RL (RL-RUM) model proposed in Fosgerau et al. (2013a), a linear-in-parameters utility is associated with each link pair in the network, and is the sum of a deterministic and a random term. A traveler maximizes his/her utility, defined as the sum of the instantaneous link utility at the current decision stage and the expected maximum utility from the sink node of outgoing links to the destination. The random terms are assumed to be i.i.d. extreme value type I, so the choice model at each stage is MNL, leading to the fact that the expected maximum utilities can be computed by solving a system of linear equations. In the following we present RL model based on the RRM decision rule. The derivation is similar to Fosgerau et al. (2013a) but the link costs and expected costs are different.

A directed connected graph (not assumed acyclic) $\mathcal{G} = (\mathcal{A}; \mathcal{V})$ is considered, where $\mathcal{A}$ and $\mathcal{V}$ are the set of links and nodes, respectively. For each link $k \in \mathcal{A}$, we denote the set of outgoing links from the sink node of $k$ by $A(k)$. We extend the network with a dummy link $d$, without successors, per destination, that is, an absorbing state. The set of all links for a given destination is hence $\bar{\mathcal{A}} = \mathcal{A} \cup \{d\}$. Given two links $a, k \in \bar{\mathcal{A}}$, $a \in A(k)$, we associate the following
instantaneous random regret for individual \( n \)

\[ rr_n(a|k) = r_n(a|k) + \mu \epsilon_n(a) \]

where \( r_n(a|k) \) is the deterministic part of the link regret associated with link \( a \) given \( k \), \( -\epsilon_n(a) \) are i.i.d. extreme value type I distributed error terms and \( \mu \) is a strictly positive scale parameter. We ensure that \( \epsilon_n(a) \) have zero mean by subtracting Euler’s constant. For notational simplicity, weomit from now on the index for individual \( n \) but note that the regrets and random terms can be individual specific.

At each current state \( k \) the traveler observes the realizations of the random terms \( \epsilon(a), a \in A(k) \). He/she then chooses link \( a \) that minimizes the sum of instantaneous random regret \( rr(a|k) \) and expected downstream regret. The latter, denoted by \( R^d(k) \), is defined as the expected minimum regret from state \( k \) to the destination (see Figure 6.1). The superscript \( d \) indicates that the expected minimum regrets are destination specific (through dummy link \( d \)). \( R^d(k) \) is recursively defined by Bellman’s equation as

\[ R^d(k) = \mathbb{E} \left[ \min_{a \in A(k)} \left\{ r(a|k) + R^d(a) + \mu \epsilon(a) \right\} \right], \quad \forall k \in A. \quad (6.4) \]

We note that \( R^d(k) \) and \( r(a|k) \) may be conditional on the model parameters so they can be written as \( R^d(k) = R^d(k; \beta) \) and \( r(a|k) = r(a|k; \beta) \) where \( \beta \) is the vector of parameters to estimate. We however omit \( \beta \) for notational simplicity. Equation (6.4) can be written as

\[ R^d(k) = \mathbb{E} \left[ - \max_{a \in A(k)} \left\{ -r(a|k) - R^d(a) - \mu \epsilon(a) \right\} \right] \\
= -\mathbb{E} \left[ \max_{a \in A(k)} \left\{ -r(a|k) - R^d(a) + \mu (-\epsilon(a)) \right\} \right], \quad \forall k \in A, \]

or equivalently

\[ \frac{1}{\mu} R^d(k) = -\mathbb{E} \left[ \max_{a \in A(k)} \left\{ \frac{1}{\mu} (-r(a|k) - R^d(a)) + (-\epsilon(a)) \right\} \right], \quad \forall k \in A \quad (6.5) \]
Since $-\epsilon(a)$ are i.i.d. standard extreme value type I by assumption, the probability of choosing link $a$ given $k$ is given by the MNL model

$$P^d(a|k) = \frac{\xi(a|k)e^{-\frac{1}{\mu}(r(a|k)+R^d(a))}}{\sum_{a'\in A(k)}e^{-\frac{1}{\mu}(r(a'|k)+R^d(a'))}}, \quad \forall a, k \in \tilde{A}. \quad (6.6)$$

Note that we include $\xi(a|k)$ that equals one if $a \in A(k)$ and zero otherwise so that the probability is defined for all $a, k \in \tilde{A}$ (we recall that $\tilde{A} = A \cup \{d\}$). Since the choice model at each state is MNL, the expected minimum regrets are given recursively by the logsum

$$-\frac{1}{\mu}R^d(k) = \mathbb{E}\left[\max_{a \in A(k)} \left\{\frac{1}{\mu}(-r(a|k) - R^d(a)) + (-\epsilon(a))\right\}\right]$$

$$= \ln \left(\sum_{a \in A(k)} e^{\frac{1}{\mu}(-r(a|k)-R^d(a))}\right), \forall k \in A, \quad (6.7)$$

and $R^d(d) = 0$ by assumption. We define a matrix $M^d$ of size $|\tilde{A}| \times |\tilde{A}|$ and a vector $z$ of size $|\tilde{A}|$ with entries

$$M^d_{ka} = \xi(a|k)e^{-\frac{1}{\mu}r(a|k)}, \quad z^d_{k} = e^{-\frac{1}{\mu}R^d(k)}, \quad \forall k, a \in \tilde{A}. \quad (6.8)$$

From (6.7) we have

$$z^d_k = \begin{cases} \sum_{a \in A} M^d_{ka} z^d_a & \text{if } k \in A \\ 1 & \text{if } k = d, \end{cases} \quad (6.9)$$

and the system in (6.9) can be written in matrix form as

$$z^d = M^d z^d + b,$$

or equivalently,

$$z^d = (I - M^d)^{-1}b, \quad (6.10)$$

where $b$ is a vector of size $|\tilde{A}|$ with zeros values for all states except for the destination $d$ that equals 1 and $I$ is the identity matrix. Similar to Fosgerau et al. (2013a) we obtain a system of linear equations which can be solved in short computational time.

Fosgerau et al. (2013a) discuss the existence of a solution to Bellman’s equation for the RL-RUM model and this can be applied in the context of the RRM-based RL models. In essence, the existence of a solution depends on the size of the scaled instantaneous regrets and on the balance between the number of paths connecting the nodes in the network. It is easy to find a feasible solution by increasing the magnitude of the model parameters. Note that if the scales $\mu$ are different over links, the system in (6.9) becomes nonlinear, similar to the one given in Mai et al. (2015c).
Using (6.6), the probability of choosing link \( a \) given a state \( k \) can be written as

\[
P^d(a|k) = \xi(a|k)e^{-\frac{1}{\mu}(r(a|k) + Rd(a) - Rd(k))}, \quad \forall k, a \in \tilde{A},
\]

and the probability of a path defined by a sequence of links \( \sigma = [k_0, \ldots, k_J] \) is

\[
P^d(\sigma) = \prod_{i=0}^{J-1} P^d(k_{i+1}|k_i) = e^{\frac{1}{\mu}Rd(k_0)} \prod_{i=0}^{J-1} e^{-\frac{1}{\mu}r(k_{i+1}|k_i)} = e^{\frac{1}{\mu}Rd(k_0)} e^{-\frac{1}{\mu}r(\sigma)},
\]

where \( r(\sigma) = \sum_{i=0}^{J-1} r(k_{i+1}|k_i) \). Given two paths \( \sigma_1 \) and \( \sigma_2 \), the ratio between two probabilities is

\[
\frac{P(\sigma_2)}{P(\sigma_1)} = e^{\frac{1}{\mu}(r(\sigma_2) - r(\sigma_1))},
\]

and it does not only depend on the attributes of links on paths \( \sigma_1, \sigma_2 \). Hence, the IIA property does not hold for the RRM models. In the next section we discuss different formulations of the instantaneous link regret functions.

### 6.4 Link regret formulations

We define the regret \( r(a|k) \) of link \( a \in A(k) \) conditional on link \( k \in A \), based on the GRRM model given by (6.3). A transportation network can have links with only one successor (\( |A_k| = 1 \)). In this case, the next-link probabilities should equal 1 and hence do not affect the path choice probabilities. Indeed, these links could be removed by preprocessing. We simply derive link regret formulations based on all the alternatives, so that a preprocessing is not necessary.

When a node has only one outgoing link, existing random regret models would assign a regret zero which could cause numerical issues for the RL model. We therefore define the regret based on all outgoing links and the slightly modified GRRM is

\[
r_{\text{GRRM}}(a|k) = \sum_{a' \in A(k)} \sum_{t} \ln \left( \lambda_t + e^{\bar{\beta}_t(x(a'|k)t - x(a|k)t)} \right), \quad \forall k \in A, a \in A(k),
\]

(6.11)

where \( x(a|k) \) is a vector of attributes associated with link \( a \) given \( k \), \( \lambda \) and \( \beta \) are vector of parameters to be estimated. The only difference here with respect to the model in Chorus (2014) is that the first sum is over all alternatives.

We also define a new formulation for regret that we call Extended Random Regret Minimization (ERRM), by adding terms associated with the attributes of the non-chosen alternatives to the attribute differences in (6.11). The ERRM has the following formulation

\[
r_{\text{ERRM}}(a|k) = \sum_{a' \in A(k)} \sum_{t} \ln \left( \lambda_t + e^{\bar{\beta}_t(x(a'|k)t - x(a|k)t) + \delta_t x(a'|k)t} \right), \quad \forall k \in A, a \in A(k).
\]

(6.12)
The difference lies in the term $\delta_t x(a'|k)_t$. If $\delta_t > 0$, the impact of the non-chosen alternatives becomes larger and if $\delta_t < 0$, it is smaller, compared to the GRRM model. Moreover, if $\delta_t = 0$ we obtain the GRRM formulation.

The regret in (6.12) can be written as

$$r^{ERRM}(a|k) = \sum_{a' \in A(k)} \sum_t \ln \left( \lambda_t + e^{-\beta_t x(a|k)_t} + (\beta_t + \delta_t) x(a'|k)_t \right), \forall k \in A, a \in A(k),$$

which clearly show that if $\lambda_t = 0$ and $\delta_t = -\beta_t$, then the regret in (6.12) is linear-in-parameters and the attributes associated with the non-chosen alternatives are excluded, that is

$$r^{ERRM}(a|k; \lambda = 0, \delta = -\beta) = -|A(k)|\beta^T x(a|k) = -|A(k)|v(a|k),$$

where $v(a|k)$ are the linear-in-parameters utilities as in Fosgerau et al. (2013a). So the regret is also linear-in-parameters but different from the RUM-based model with a factor $|A(k)|$. This factor appears because the sum in the regret formula is over all the outgoing links from the sink node of $k$. We propose an Averaged Random Regret Minimization (ARRM) model as an alternative to the ERRM where a normalization factor is used so that the regret is averaged over all the alternatives

$$r^{ARRM}(a|k) = \frac{1}{|A(k)|} r^{ERRM}(a|k), \forall k \in A, a \in A(k). \quad (6.13)$$

Accordingly, by specifying $\lambda_t = 0$ and $\delta_t = -\beta_t$, $\forall t$, we obtain $r^{ARRM}(a|k) = -v(a|k)$. Based on (6.8) the entries of matrix $M^d$ becomes

$$M^d_{ka} = \xi(a|k)e^{-\frac{1}{\mu}V^d(k)}, \forall k, a \in A.$$ 

We refer to the definition of the matrix $M^d$ in Fosgerau et al. (2013a) and note that $z^d$ is a solution to the system of linear equations $z^d = (I - M^d)^{-1}b$, therefore it is straightforward to show that

$$z^d_k = e^{\frac{1}{\mu}R^d(k)} = e^{\frac{1}{\mu}V^d(k)}, \forall k \in A,$$

where $V^d(k)$ is the expected maximum utility from state $k$ to the destination. The probability of choosing a link $a$ given link $k$ can be written as

$$P^d(a|k) = \xi(a|k)e^{-\frac{1}{\mu}(v(a|k)+R^d(a)-R^d(k))} = \xi(a|k)e^{\frac{1}{\mu}(v(a|k)+V^d(a)-V^d(k))}. $$

This choice probability is equivalent to the one given by the RL-RUM model. So the RL model based on ARRM model generalizes the RL-RUM model.
6.5 Competitive RUM models

We present three RUM RL models in order to take advantage from nonlinear-in-parameters utilities and explicitly account for the non-chosen alternatives. The purpose is to have RUM models that are as close as possible to the RRM ones to provide a fair comparison. A first model, called Competitive Random Utility Maximization (CRUM), is obtained by simply permuting the attribute terms in the regrets (6.11), leading to the deterministic utilities

\[ v^{\text{CRUM}}(a|k) = \sum_{a' \in A(k)} \sum_{t} \ln \left( \lambda_t + e^{\beta_t(x(a|k)_t - x(a'|k)_t)} \right), \forall k \in A, a \in A(k). \] (6.14)

The rationale behind this model is that the decision-maker favors alternatives with more interesting attributes compared to those of the competing alternatives. This model differs from the classical RUM framework in that the attributes differences are explicitly taken into account in the utilities. In order to make the distinction more clear, we refer to the RUM models as the linear-in-parameters ones proposed in Fosgerau et al. (2013a), and to the CRUM as the nonlinear-in-parameters RUM proposed in this section. Moreover, similarly to the ERRM and ARRM models, we also propose two extensions of the CRUM model in order to better capture the impact of non-chosen alternatives in the link utility models. The first extension, called extended CRUM (ECRUM) model, has the following form

\[ v^{\text{ECRUM}}(a|k) = \sum_{a' \in A(k)} \sum_{t} \ln \left( \lambda_t + e^{\beta_t(x(a|k)_t - x(a'|k)_t)} + \delta_t x(a'|k)_t \right), \forall k \in A, a \in A(k). \] (6.15)

The second model, called averaged CRUM (ACRUM), is obtained by averaging the utilities over all the alternatives

\[ v^{\text{ACRUM}}(a|k) = \frac{1}{|A(k)|} \sum_{a' \in A(k)} \sum_{t} \ln \left( \lambda_t + e^{\beta_t(x(a|k)_t - x(a'|k)_t)} + \delta_t x(a'|k)_t \right), \forall k \in A, a \in A(k). \] (6.16)

Indeed, by specifying \( \lambda_t = 0 \) and \( \beta_t = \delta_t \) we have \( v^{\text{ACRUM}}(a|k; \lambda = 0, \delta = \beta) = v(a|k), \forall k \in A, a \in A(k), \) so the ACRUM also generalizes the basic RUM models specified in Fosgerau et al. (2013a).

6.6 Illustrative example

In this section, similar to several studies in the literature (for instance Ben-Akiva and Bierlaire, 1999a, Mai et al., 2015c), we use the simple three path network shown in Figure 6.2 to illustrate how the RRM and CRUM RL models relaxes the IIA property. The network consists of three paths connecting link \( o \) (origin) and dummy link \( d \) (destination), namely \([o, a, d], [o, b, c, d],\)
[o, c, e, d]. We number these paths 1, 2 and 3 and the corresponding path probabilities are \( P_1 \), \( P_2 \) and \( P_3 \), respectively. Only a link length attribute is included in the deterministic regrets and the values are given in the parentheses on each arc.

\[
\begin{align*}
o & \rightarrow a & a(4) & \rightarrow d(0) \\
& \rightarrow c & c(1) & \rightarrow 0 \\
& \rightarrow b & b(1) & \rightarrow e & e(3)
\end{align*}
\]

**Figure 6.2:** Classic three paths example network

First, we note that both the RL-RUM and path-based MNL RUM and RRM models assign the probability \( 1/3 \) to the three paths. The path size logit model (Ben-Akiva and Bierlaire, 1999b) may assign a probability of \( 1/2 \) to path \([o, a, d]\) and a probability of \( 1/4 \) to paths \([o, c, e, d]\) and \([o, b, e, d]\). We illustrate the choice probabilities given by the GRRM, ERRM, ARRM, CRUM, ECRUM and ACRUM RL models in Figure 6.3. We vary \( \lambda \) over the interval \([0, 1]\) for these models. For the GRRM and CRUM models, \( \beta \) varies over \([-2, 3]\) and for the ERRM, ARRM, ECRUM and ACRUM models we vary \( \delta \) over the interval \([-4, 4]\) while keeping \( \beta \) fixed to \(-1\). The intervals and values of the parameters \( \beta \), \( \lambda \) and \( \delta \) are chosen manually with the purpose of showing how the choice probabilities change when the parameters vary.

We note that the path probabilities are indeed no-longer equal. The path probabilities given by the GRRM are similar to those given by the CRUM when \( \lambda \) and \( \beta \) vary. On the contrary, the ERRM and ARRM models performs differently from the ECRUM and ACRUM. For the ERRM and ARRM models, if the value of \( \delta \) is small, \( P_2 \) and \( P_3 \) are close to \( 1/2 \) and \( P_1 \) approaches zero. On the contrary, for the ECRUM and ACRUM models, \( P_2 \) and \( P_3 \) are close to \( 1/2 \) and \( P_1 \) is close to 0 if the value of \( \delta \) becomes large. Recall that the ARRM and ACRUM become the RUM model when \( \lambda = 0 \) and \( \delta = -\beta \) (for the ARRM) or \( \delta = \beta \) (for the ACRUM). This is indeed shown in Figure 6.4. Moreover, the results from this example suggest that, compared to other parameters, the impact of parameter \( \lambda \) on the path probabilities is small for the ARRM and ERRM models, but significant for the ACRUM and ECRUM.

Finally, we note that the regrets given by the RRM and CRUM models are nonlinear functions of the model parameters. This leads to more complex path probabilities than RUM models. It is clear from the numerical example in Figure 6.4 showing plots of \( P_1 \) as functions of \( \beta \) and \( \delta \). This may lead to the fact that RRM and CRUM models are more complicated and time consuming to estimate than RUM models where the link utilities are linear-in-parameters, which is confirmed by our numerical results.
There are different ways of estimating a dynamic discrete choice model (see for instance Aguirregabiria and Mira, 2010). Similar to Fosgerau et al. (2013a) and Mai et al. (2015c) we use the nested fixed point algorithm proposed by Rust (1987). This algorithm combines an outer iterative nonlinear optimization algorithm for searching over the parameter space with an inner algorithm for solving the expected minimum regrets or the expected maximum utilities (or the value functions). The value functions can be solved quickly using the system of linear equations in (6.10). The log-likelihood function for the CRUM models can be derived similarly as in

**Figure 6.3**: Path probabilities

### 6.7 Maximum likelihood estimation
Fosgerau et al. (2013a). We therefore turn our attention to the definition of the log-likelihood function as well as its derivatives for the RRM models.

The log-likelihood function defined for $N$ observations $\sigma_1, \ldots, \sigma_N$ with respect to the vector of model parameters $\beta$ is

$$LL(\beta) = \sum_{n=1}^{N} \ln P(\sigma_n) = \frac{1}{\mu} \sum_{n=1}^{N} \sum_{i=0}^{J_n-1} (R(k_i^u(n)) - r(\sigma_n)).$$

For notational simplicity we omit the superscript $d$ indicating the destinations but note that the choice probabilities $P(\sigma_n)$ and expected minimum regrets $R(k_i^u(n))$ depend on the destination of path $\sigma_n$. Efficient nonlinear techniques for the problem require analytical derivatives of the log-likelihood function. We therefore derive the gradient of $LL(\beta)$ with respect to a parameter $\beta_i$ as

$$\frac{\partial LL(\beta)}{\partial \beta_i} = \frac{1}{\mu} \sum_{n=1}^{N} \sum_{i=0}^{J_n-1} \left( \frac{\partial R(k_i^u(n))}{\partial \beta_i} - \frac{\partial r(\sigma_n)}{\partial \beta_i} \right),$$

which requires the derivatives of $R(k_i^u(n))$. Taking the first derivative of (6.10), we obtain

$$\frac{\partial z}{\partial \beta_i} = (I - M)^{-1} \frac{\partial M}{\partial \beta_i} z,$$

and using $\frac{\partial R(k)}{\partial \beta_i} = -\mu \frac{\partial z_k}{z \partial \beta_i}$. 

(6.17)
The gradients of the regret value function $R(k)$, $k \in \bar{A}$, can be quickly computed using the system of linear equations (6.17). The value of $r(\sigma)$ for a given path $\sigma$ is nonlinear-in-parameters, so that $\frac{\partial r(\sigma)}{\partial \beta_i}$ has a complicated form but is easy to derive. We note that from (6.11), (6.12) and (6.13) the regret-based models have three vector of parameters to be estimated i.e. $\lambda$, $\beta$ and $\delta$. The GRRM model requires $0 \leq \lambda_t \leq 1$ for all attributes $t$. This implies that the MLE becomes a constrained optimization problem as in the following

$$\max_{\lambda, \beta, \delta} \quad LL(\lambda, \beta, \delta).$$

It is important to note that the RRM and CRUM models require comparisons between each alternative with every other one, attribute per attribute. This makes the estimation of the RRM and CRUM models more costly, compared to the classical RUM models in Fosgerau et al. (2013a). In order to simplify the estimation, we apply the decomposition (DeC) method (Mai et al., 2015a) designed to speed up the estimation of RL models. This approach requires to solve one linear system only when computing the log-likelihood function, instead of solving one system of linear equations for each observed destination or origin-destination pair. The drawback of this method is that it is not compatible with the link size (LS) attribute (for instance Fosgerau et al., 2013a) and the NRL model (Mai et al., 2015c).

Finally, we use the interior point algorithm with BFGS to solve this constrained problem. The code is implemented in MATLAB (available upon request) and we use the function *fmincon* for solving the problem.

### 6.8 Numerical results

In order to have comparable numerical results with previous studies, we use the same data as Fosgerau et al. (2013a) (also used in Frejinger and Bierlaire, 2007, Mai et al., 2015c,d), collected in the city of Borlänge, Sweden. This network is composed of 3077 nodes and 7459 links and it is uncongested so travel times are assumed static and deterministic. There are 1832 observations containing 466 destinations, 1420 different origin-destination (OD) pairs and more than 37,000 link choices. Moreover, as we explained in the following we specify the link regret functions using the same attributes as Fosgerau et al. (2013a) and Mai et al. (2015c).

#### 6.8.1 Model specifications

Four attributes are included in the regret function: travel time $TT(a)$ of link $a$, number of left turn $LT(a|k)$ that equals one if the turn angle from $k$ to $a$ is larger than 40 degrees and less
than 177 degrees, link constant $LC(a)$ that equals one except the dummy link which equals zero and U-turn $UT(a|k)$ that equals one if the turn angle is larger than 177.

For the sake of comparison, we report the estimation and prediction results for the RUM-based RL (Fosgerau et al., 2013a) and NRL (Mai et al., 2015c) models, their deterministic utility specifications are

$$v^{RL}(a|k; \beta) = v^{NRL}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k),$$

$$v^{RL-LS}(a|k; \beta) = v^{NRL-LS}(a|k; \beta) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k) + \beta_{LS}LS(a),$$

where $LS$ is the link size attribute (for a detailed description see Fosgerau et al., 2013a). It has been computed using a linear-in-parameters formulation of the aforementioned four attributes using parameters $\tilde{\beta}_{TT} = -2.5$, $\tilde{\beta}_{LT} = -1$, $\tilde{\beta}_{LC} = 0.4$, $\tilde{\beta}_{UT} = -4$. This attribute can be considered as a correction for the utilities in order to relax with the IIA property from the RL model, similar to the path size attribute (Ben-Akiva and Bierlaire, 1999a). The NRL model has the same instantaneous utility but the IIA is relaxed by allowing the random terms to have link specific scale parameters.

The regret specifications for the RRM models can be defined based on (6.11), (6.12) and (6.13), respectively, using the same four attributes as the RUM models. As mentioned earlier, the DeC method (Mai et al., 2015a) is used to estimate the resulting expensive estimation problems in reasonable time, and the LS attribute and the NRL model are not compatible with the DeC method. We therefore only estimate the RRM and CRUM models based on the RL with the four aforementioned attributes.

There is an important difference related to the $LC$ attribute. In the RUM model, the rationale behind using $LC(a)$ in the instantaneous utilities is to penalize paths with many crossings (links). In the regret context, the link constant equals one except for the dummy link which equals zero. So this attribute cancels out when comparing two outgoing links except when comparing a link in $A$ with dummy link $d$. More precisely, for each link $k \in A$, the regret for the ERRM model can be expressed as

$$r^{ERRM}(a|k) = \sum_{l,t \neq LC} \sum_{a' \in A(k)} \left( \lambda_t + e^{\gamma_l x(a'|k) - x(a|k)}) + \delta'_t x(a'|k) + \right) + \psi(a|k)_{LC}, \quad (6.18)$$
where

\[
\psi(a|k)_{LC} = \begin{cases} 
\sum_{a' \in A(k)} \ln(\lambda_{LC} + e^{\delta_{LC}}), & \forall a \in A(k), a \neq d, d \notin A(k) \\
\sum_{a' \in A(k)} \ln(\lambda_{LC} + e^{\delta_{LC}}) + \ln(\lambda_{LC} + e^{-\beta_{LC} + \delta_{LC}}), & \text{if } d \in A(k), a \neq d \\
\sum_{a' \in A(k)} \ln(\lambda_{LC} + e^{\beta_{LC} + \delta_{LC}}), & \text{if } a = d.
\end{cases}
\]

Equations (6.18) and (6.19) indicate that the value of \(\beta_{LC}\) only affects the regret \(r^{ERRM}(a|k)\) if link \(k\) connects directly to dummy link \(d\). The other RRM-based regrets and CRUM-based utilities can be written in a similar way. Consequently, the link constant in the RRM and CRUM models plays a different role from the one in the RUM models; it is an attraction factor at the destination. Such a factor is actually important for the RRM and CRUM models to ensure that the probability of choosing the destination link (once arriving at the destination) is close to one. Such an attraction attribute is not needed (and does not affect the probabilities) in the RUM models since the instantaneous utilities are negative except for the dummy link that is zero. In order to make the distinction clear between these attributes, we call it destination constant (DC) in the RRM and CRUM models. Accordingly, the regrets for the three RRM models are

\[
r_{ERRM}(a|k) = \sum_{a' \in A(k)} \left\{ \ln(\lambda_{TT} + e^{\beta_{TT}(TT(a')-TT(a))+\delta_{TT}TT(a')}) \\
+ \ln(\lambda_{LT} + e^{\beta_{LT}(LT(a'|k)-LT(a|k))+\delta_{LT}LT(a'|k)}) \\
+ \ln(\lambda_{DC} + e^{\beta_{DC}(DC(a')-DC(a))+\delta_{DC}DC(a')}) \\
+ \ln(\lambda_{UT} + e^{\beta_{UT}(UT(a'|k)-UT(a|k))+\delta_{UT}UT(a'|k)}) \right\} \\
\forall k \in \mathcal{A}, a \in A(k),
\]

\[
r_{ARRM}(a|k) = \frac{1}{|A(k)|} r_{ERRM}(a|k), \forall k \in \mathcal{A}, a \in A(k),
\]

\[
r_{GRRM}(a|k) = \sum_{a' \in A(k)} \left\{ \ln(\lambda_{TT} + e^{\beta_{TT}(TT(a')-TT(a))}) \\
+ \ln(\lambda_{LT} + e^{\beta_{LT}(LT(a'|k)-LT(a|k)}) \\
+ \ln(\lambda_{DC} + e^{\beta_{DC}(DC(a')-DC(a))}) \\
+ \ln(\lambda_{UT} + e^{\beta_{UT}(UT(a'|k)-UT(a|k)}) \right\}, \\
\forall k \in \mathcal{A}, a \in A(k).
\]
The utilities for the CRUM models also can be specified similarly as

\[ v^\text{ECRUM}(a|k) = \sum_{a' \in A(k)} \left\{ \ln(\lambda_{TT} + e^{\beta_{TT}(TT(a) - TT(a'))} + \delta_{TT}TT(a')) + \ln(\lambda_{LT} + e^{\beta_{LT}(LT(a|k) - LT(a'|k))} + \delta_{LT}LT(a'|k)) + \ln(\lambda_{DC} + e^{\beta_{DC}(DC(a) - DC(a'))} + \delta_{DC}DC(a')) + \ln(\lambda_{UT} + e^{\beta_{UT}(UT(a) - UT(a'|k))} + \delta_{UT}UT(a'|k)) \right\} \]

\[ \forall k \in A, a \in A(k), \]

\[ v^\text{ACRUM}(a|k) = \frac{1}{|A(k)|} v^\text{ECRUM}(a|k), \forall k \in A, a \in A(k), \]

\[ v^\text{CRUM}(a|k) = \sum_{a' \in A(k)} \left\{ \ln(\lambda_{TT} + e^{\beta_{TT}(TT(a) - TT(a'))}) + \ln(\lambda_{LT} + e^{\beta_{LT}(LT(a|k) - LT(a'|k)))} + \ln(\lambda_{DC} + e^{\beta_{DC}(DC(a) - DC(a'))} + \ln(\lambda_{UT} + e^{\beta_{UT}(UT(a) - UT(a'|k))}) \right\}, \]

\[ \forall k \in A, a \in A(k). \]

### 6.8.2 Estimation results

The estimation results for the three RRM models and the two extensions of the CRUM model (i.e. ECRUM and ACRUM) are presented in Table 6.1. We do not estimate the CRUM model because we could not find a feasible initial vector of parameters for this model (parameters \( \lambda \) and \( \beta \) such that the corresponding Bellman’s equation has a solution). (6.14) suggests that the link utilities could be positive or negative. Since the network contains cycles, expected maximum utilities (or value functions) may take infinite values, and Bellman’s equation in this case cannot be solved. On the contrary, for the other models, it is easy to find feasible parameters \( \beta \) by specifying \( \lambda = 0, \beta = -\delta \) for the RRM models, and \( \beta = \delta \) for the CRUM models. This specification allows to exclude the impact of non-chosen alternatives in the link utilities or link regrets, so that the utilities and regrets have linear-in-parameters formulas, and are close to the utilities specified in Fosgerau et al. (2013a).

Before discussing the estimation results we note that even the CRUM models have similar forms as the RRM ones, their numerical results are different. This difference can be explained by the fact that these models are based on the maximums versus minimums of highly nonlinear and asymmetric functions.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>GRRM</th>
<th>ARRM</th>
<th>ERRM</th>
<th>ACRUM</th>
<th>ECRUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\beta}_{TT}$</td>
<td>-0.15</td>
<td>-1.92</td>
<td>-0.37</td>
<td>-1.178</td>
<td>-0.13</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.01</td>
<td>0.21</td>
<td>0.09</td>
<td>0.18</td>
<td>0.19</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-11.46</td>
<td>-8.98</td>
<td>-4.05</td>
<td>-6.33</td>
<td>-0.70</td>
</tr>
<tr>
<td>$\hat{\beta}_{LT}$</td>
<td>-0.34</td>
<td>-1.8</td>
<td>-0.31</td>
<td>-0.97</td>
<td>-0.69</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.02</td>
<td>0.41</td>
<td>0.08</td>
<td>0.59</td>
<td>0.60</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-15.36</td>
<td>-4.43</td>
<td>-3.84</td>
<td>-1.63</td>
<td>-1.5</td>
</tr>
<tr>
<td>$\hat{\beta}_{UT}$</td>
<td>-5.89</td>
<td>-7.32</td>
<td>-5.32</td>
<td>-4.38</td>
<td>-17.34</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.57</td>
<td>65.32</td>
<td>1.87</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-10.32</td>
<td>-0.11</td>
<td>-2.85</td>
<td>-5.08</td>
<td>-20.07</td>
</tr>
<tr>
<td>$\hat{\beta}_{DC}$</td>
<td>12.92</td>
<td>99.99</td>
<td>23.18</td>
<td>-0.67</td>
<td>-1.92</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>1.66</td>
<td>36.03</td>
<td>3.79</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>7.77</td>
<td>2.77</td>
<td>6.11</td>
<td>-0.88</td>
<td>-2.51</td>
</tr>
<tr>
<td>$\hat{\delta}_{TT}$</td>
<td>-</td>
<td>3.75</td>
<td>1.22</td>
<td>-3.49</td>
<td>-0.75</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.46</td>
<td>0.21</td>
<td>0.48</td>
<td>0.48</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>8.18</td>
<td>5.69</td>
<td>-7.31</td>
<td>-1.58</td>
</tr>
<tr>
<td>$\hat{\delta}_{LT}$</td>
<td>-</td>
<td>0.12</td>
<td>0.09</td>
<td>-0.69</td>
<td>-0.29</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.7</td>
<td>0.1</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>0.17</td>
<td>0.89</td>
<td>-2.17</td>
<td>-0.89</td>
</tr>
<tr>
<td>$\hat{\delta}_{UT}$</td>
<td>-</td>
<td>7.16</td>
<td>4.75</td>
<td>-4.23</td>
<td>-16.64</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>53.66</td>
<td>1.31</td>
<td>1.05</td>
<td>1.05</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>53.66</td>
<td>1.31</td>
<td>1.05</td>
<td>1.05</td>
</tr>
<tr>
<td>$\hat{\delta}_{DC}$</td>
<td>-</td>
<td>-7.16</td>
<td>-4.4</td>
<td>-0.32</td>
<td>-20.53</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>63.49</td>
<td>1.73</td>
<td>5.17</td>
<td>5.17</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>-63.49</td>
<td>1.73</td>
<td>5.17</td>
<td>5.17</td>
</tr>
<tr>
<td>$\hat{\lambda}_{TT}$</td>
<td>8.13E-06</td>
<td>0.37</td>
<td>1.00</td>
<td>2.30E-06</td>
<td>5.33E-06</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.31</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>1.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\hat{\lambda}_{LT}$</td>
<td>7.26E-06</td>
<td>1.00</td>
<td>8.29E-05</td>
<td>4.50E-06</td>
<td>1.00</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>-</td>
<td>0.31</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-</td>
<td>1.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\hat{\lambda}_{UT}$</td>
<td>0.76</td>
<td>0.01</td>
<td>1.04E-04</td>
<td>2.10E-06</td>
<td>0.26</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.04</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.03</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>17.86</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8.75</td>
</tr>
<tr>
<td>$\hat{\lambda}_{DC}$</td>
<td>0.46</td>
<td>0.58</td>
<td>0.48</td>
<td>0.00</td>
<td>0.28</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.02</td>
<td>37.06</td>
<td>0.81</td>
<td>-</td>
<td>0.22</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>21.62</td>
<td>0.02</td>
<td>0.59</td>
<td>-</td>
<td>1.28</td>
</tr>
</tbody>
</table>

Table 6.1: Estimation results

For the RRM models, the $\hat{\beta}$ are significantly different from zero except for the parameter associated with u-turns in the ARRM model. Moreover, they are, as expected, negative for travel time, left turns and u-turns. Based on the discussion in the previous section, it is also expected that $\hat{\beta}_{DC}$ are positive and with large magnitudes so that $P(d|k)$ are close to one. For the ECRUM and ACRUM models, all the $\hat{\beta}$ have their expected signs, and they are only significantly different from zero for the parameters associated with travel time, u-turns in the ACRUM model, and for the estimates associated with u-turns, destination constant in the ECRUM model.

We now turn our attention to the $\delta$ estimates. Note that the parameters $\delta$ are designed for the ERRM, ARRM, ECRUM and ACRUM models to flexibly capture the impact of non-chosen alternatives in the utilities or regrets. If $\hat{\delta}_t > 0$, the impact of non-chosen alternatives is larger than if $\hat{\delta}_t < 0$. For the RRM model, the estimation results show that $\hat{\delta}_t$ are either not
significantly different from zero, or they are significant and positive ($\hat{\delta}_{TT}$ in the ARRM and $\hat{\delta}_{UT}$ in the ERRM model). It means that the impacts of the non-chosen alternatives in the ERRM is larger than the GRRM model in terms of travel time and u-turns. On the contrary, for the CRUM models, $\delta$ estimates are not significant, or significant and negative. So the $\delta$ parameters in the CRUM models capture the opposite impacts of those from the RRM models.

We recall that if $\lambda_t = 0$ the regret associated with attribute $t$ is linear-in-parameters and if $\lambda_t = 1$ the regret becomes the original RRM model proposed by Chorus (2012). The four last rows of Table 6.1 show the $\lambda$ estimates. We do not provide standard errors and $t$-tests for the estimates that are on the bounds (close to 0 or 1) since the respective gradient values are not close to zero. For the GRRM model, the $\hat{\lambda}_t$ are only significantly different from zero for the parameters associated with u-turns and destination constant. The others are very to zero (on the bound). However, for the ERRM or ARRM models, the $\hat{\lambda}_t$ are either on the bounds ($\hat{\lambda}_{LT}$, $\hat{\lambda}_{UT}$ for the ARRM and $\hat{\lambda}_{TT}$, $\hat{\lambda}_{LT}$, $\hat{\lambda}_{UT}$ for the ERRM), or not significantly different from zero ($\hat{\lambda}_{TT}$, $\hat{\lambda}_{DC}$ for the ARRM and $\hat{\lambda}_{DC}$ for the ERRM). For the ACRUM model, the $\lambda$ estimates are not significant for all the attributes. For the ECRUM, the $\hat{\lambda}_t$ are on the bounds for travel time and left turn, significant for u-turns but not significant for destination constant. Chorus (2014) provide more detailed discussions on how the regrets change when parameters $\lambda$ vary in the interval $[0, 1]$. These discussions can be generally applied for the CRUM models.

We report the final log-likelihood values in Tables 6.2 and 6.3. In general, the differences in in-sample fit between the RUM, RRM and CRUM models cannot be statistically compared with a likelihood ratio test since they are not nested. Within the RUM framework, the NRL models have significantly better fit than the other RL models, and the models with LS attribute are better than the ones without. Among the RRM models, the ERRM performs better than the GRRM. Moreover, since the RL-RUM model is a restricted model of the ARRM and ACRUM models, the results show that the ARRM and ACRUM have significantly better fit than the RL-RUM. Finally, we note that the ERRM has the highest and the GRRM has the lowest final log-likelihood value.

<table>
<thead>
<tr>
<th># parameters</th>
<th>RL</th>
<th>RL-LS</th>
<th>NRL</th>
<th>NRL-LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final log-likelihood</td>
<td>-6303.9</td>
<td>-6045.6</td>
<td>-6187.9</td>
<td>-5952.0</td>
</tr>
</tbody>
</table>

Table 6.2: Final log-likelihood values of the RUM-based RL and NRL models

<table>
<thead>
<tr>
<th># parameters</th>
<th>ACRUM</th>
<th>ECRUM</th>
<th>GRRM</th>
<th>ARRM</th>
<th>ERRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final log-likelihood</td>
<td>-6162.8</td>
<td>-6177.1</td>
<td>-7931.6</td>
<td>-5661.6</td>
<td>-5500.4</td>
</tr>
</tbody>
</table>

Table 6.3: Final log-likelihood values of the RL models with the RRM and CRUM frameworks

Before discussing the out-of-sample fit of these models in the following section, we make some remarks about the computational time for the estimation. The RRM and CRUM models are
more expensive to estimate than the RUM models due to the costs to compute the link regrets/utilities, as at each choice stage of the RRM- or CRUM models, each alternative has to be compared with every alternative in the corresponding choice set. In the following we discuss the computational times for the estimation of the considered models, performed by means of a non parallelized MATLAB code running under a core i5, Intel(R) 3.20GHz machine with a x64-based processor.

Recall first that the RUM models are referred to as the linear-in-parameters RUM models presented in Fosgerau et al. (2013a) and Mai et al. (2015c), and the CRUM models are the nonlinear-in-parameters ones proposed in Section 6.5. In the RRM and CRUM models, given a vector of parameters, we need approximately 45 seconds to compute the links regrets/utilities, while for the RUM models it takes just less than 1 second as we can use matrix operations to compute linear-in-parameters utilities. As a result, we would need approximately 4 hours to compute the log-likelihood function in our experiments without the DeC approach, as many linear systems have to be solved. On the opposite, this computation only requires few minutes with the DeC method, as one system of linear equations only is required to evaluate the log-likelihood function. For the RUM models, the computation of the log-likelihood function takes approximately 5 minutes if not using the DeC, otherwise it takes just less than 10 seconds.

Moreover, the estimation of the RRM and CRUM models is also expensive due to the nonlinearities in the utilities/regrets. The nonlinear optimization algorithm needs approximately 30 iterations to converge for the RUM models while from 300 to 800 iterations are needed for the RRM and CRUM models. So, with the DeC method, the RRM and CRUM models can be estimated in approximately one day; and without the DeC method, the estimation would take several weeks. This explains why the LS attribute and the NRL model are too expensive to use with the RRM and CRUM frameworks.

6.8.3 Prediction results

In this section we report results from a cross-validation study. The objective is to compare the out-of-sample fit of the models which is useful to detect over-fitting and assess prediction performance.

Similar to Mai et al. (2015c), the sample of observations is repeatedly divided into two sets by uniformly drawing observations: one set contains 80% of the observations and is used for estimation and the other (20%) is used as holdout samples to evaluate the predicted probabilities by applying the estimated model. We generate 40 holdout samples of the same size by reshuffling the real sample and use the log-likelihood loss as the loss function to evaluate the prediction performance.
For each holdout sample \( i, 0 \leq i \leq 40 \) we estimate the parameters \( \beta^i, \delta^i \) and \( \lambda^i \) of the corresponding training sample and these parameters are used to compute the test errors \( err_i \)

\[
err_i = -\frac{1}{|PS_i|} \sum_{\sigma_j \in PS_i} \ln P(\sigma_j, \hat{\beta}^i, \hat{\delta}^i, \hat{\lambda}^i),
\]

where \( PS_i \) is the size of the prediction sample \( i \). We then compute the average of \( err_i \) over samples in order to have unconditional test error values

\[
\overline{err}_p = \frac{1}{p} \sum_{i=1}^{p} err_i \quad \forall 1 \leq p \leq 40.
\] (6.20)

For comparison we also report the predictions performances of the four RUM models.

We apply the cross-validation study for all the RUM-based RL and NRL models, and the RRM- and CRUM-based RL models proposed in this papers. Moreover, as mentioned earlier, if we specify \( \lambda = 0 \) and \( \beta = \delta \) (for CRUM models) or \( \beta = -\delta \) (for RRM models), then the ERRM and ECRUM are equivalent to linear-in-parameters RUM models in which the utilities \( v(a|k), a \in A(k) \), are multiplied with \( |A(k)| \). So, for the sake of comparison, we also apply the cross-validation for the RUM-based NRL-LS model, i.e., the best RUM model in prediction (for instance Mai et al., 2015c), where the link utilities are scaled as \( |A(k)|v(a|k), a \in A(k) \). We denote this RUM model by NRL-LS(*).

The values of \( \overline{err}_p, 1 \leq p \leq 40 \), for all the models are plotted in Figures 6.5. As expected, the value of \( \overline{err}_p \) for each model stabilizes as \( p \) increases. The results show that the ERRM model performs best (lowest value of the loss function). The performance of the ERRM model is very different from GRRM that has the worst performance. Interestingly, the ARRM has a final log-likelihood value (in-sample fit) that is almost 300 units better than the best RUM model (NRL-LS) but the prediction performance is worse than both NRL-LS and RL-LS. The ACRUM and ECRUM have a similar performance. They perform better than the RL model, but worse than the RL with the LS attribute. Moreover, the NRL-LS(*) model performs worse than all the other models except the GRRM. Since the ECRUM and ERRM perform better than the RUM-based RL, this remark indicates the important role of the non-chosen alternatives in the RRM and CRUM frameworks.

We note that all the RRM models considered in this paper are based on the sums over all alternatives due to the fact that there can be only one outgoing at a node, which could cause numerical issues in the RL model. This is, however, not the case for the Borlänge network.

<table>
<thead>
<tr>
<th>Model</th>
<th>RL</th>
<th>RL-LS</th>
<th>NRL</th>
<th>NRL-LS</th>
<th>NRL-LS(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>3.39</td>
<td>3.25</td>
<td>3.36</td>
<td>3.20</td>
<td>3.56</td>
</tr>
</tbody>
</table>

Table 6.4: Average of out-of-sample error values for the RL and NRL models
Table 6.5: Average of out-of-sample error values for the RRM and CRUM models

<table>
<thead>
<tr>
<th></th>
<th>GRRM</th>
<th>ARRM</th>
<th>ERRM</th>
<th>ACRUM</th>
<th>ECRUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4.46</td>
<td>3.31</td>
<td>3.00</td>
<td>3.33</td>
<td>3.34</td>
</tr>
</tbody>
</table>

6.9 Conclusion

In this paper, we have compared estimation results and prediction performance between the RUM, RRM and CRUM RL route choice models. We adapted the GRRM model proposed by Chorus (2014) and proposed two variants: ARRM and ERRM models. For the sake of comparison, we also proposed three nonlinear-in-parameters RUM models, which account for the non-chosen alternatives in the utilities. We provided numerical results and a cross-validation study using real data and a network of more than 3000 nodes and 7000 links. The cross-validation results indicated that the ERRM has the best in- and out-of-sample fit and the GRRM has the worst fit. These results indicate the importance of the additional terms associated with the non-chosen alternatives added to the attribute differences of the regrets.

The numerical results indicate that RRM rule may be an interesting avenue for route choice modeling. Moreover, the results from the ARRM and ACRUM models (the in-sample and out-of-sample fit is better than the RUM models) show that nonlinear utility specifications should be investigated in more detail for RUM RL models. The superiority of the ERRM and ARRM models, compared to the ECRUM and ACRUM, suggests that accounting for the non-chosen alternatives works better for the regret decision rule, at least in our experiments. This also suggests that it may be necessary to use other data sets in order to access if it is always the case for other transport networks. We note that other specifications of the RRM models have been recently proposed (see for instance van Cranenburgh et al., 2015), which could be interesting to investigate.

It is important to note that the estimation and application of the RRM and CRUM models are more complicated and time consuming, compared to the RUM ones. Moreover, as mentioned earlier, the RRM and CRUM frameworks are too costly to use with the LS attribute and the nesting idea (Mai et al., 2015c). In addition, the interpretation of the parameter estimates is less straightforward.
Figure 6.5: Average of the test error values over holdout samples
Chapter 7

A Misspecification Test for Logit Based Route Choice Models

This chapter concerns a misspecification test for route choice models. We show how the information matrix test proposed by White (1982) can be applied to test two MNL route choice models, i.e. path-based models under sampling of alternatives (Frejinger et al., 2009), and the RL models. First, we show the validity of the information matrix equality for the path-based models, so that the information matrix test can be applied. We then present a Monte Carlo experiment using simulated data to assess the size and the power of the test and to compare its performance with two other tests in the literature. We also test the models estimated on real data and we reject the hypothesis that these models are correctly specified. The results on real data imply that alternative model structures with correlated random terms, such as the NRL (Chapter 3), RCNL (Chapter 4) or the mixed RL (Chapter 5) models, should be investigated.

A preliminary version of this paper was presented at the 3rd Symposium of the European Association for Research in Transportation (hEART), Leeds, UK, 10-12 September, 2014. The paper has been published as:

A Misspecification Test for Logit Based Route Choice Models

Tien Mai¹  Emma Frejinger ¹  Fabian Bastin ¹

Abstract

The multinomial logit (MNL) model is often used for analyzing route choices in real networks in spite of the fact that path utilities are believed to be correlated. Yet, statistical tests for model misspecification are rarely used. This paper shows how the information matrix test for model misspecification proposed by White (1982) can be applied to test path-based and link-based MNL route choice models.

We present a Monte Carlo experiment using simulated data to assess the size and the power of the test and to compare its performance with the IIA (Hausman and McFadden, 1984) and McFadden-Train Lagrange multiplier (McFadden and Train, 2000) tests. Moreover, we test models estimated on real data and they are all rejected. Attributes correcting for correlation (path size and link size) improve model fit but do not affect the outcome of the test.

Keywords: route choice; model misspecification testing; information matrix test; recursive logit; path size logit; link size

7.1 Introduction

The multinomial logit (MNL) model is often used for analyzing route choices in real networks in spite of the fact that path utilities are believed to be correlated. Yet, statistical testing of model misspecification is rarely performed. The information matrix (IM) test proposed by White (1982) is a general test for model misspecification. It exploits the well-known information matrix equality, which states that if a model is correctly specified, the expectation of the sum of the Hessian matrix and the outer product of the scores is zero. The test statistic (Theorem 4.1 in White, 1982) has a quite complicated form and contains third derivatives which raises computational concerns. In this paper we show how the IM test can be applied to path-based and link-based (recursive logit proposed by Fosgerau et al., 2013) MNL route choice models. Moreover, we assess the size and the power of the test in a Monte Carlo experiment using simulated observations. The code for estimating route choice models and applying the test has been implemented in MATLAB and is freely available upon request.

There are different specification tests available for the MNL model. Hausman and McFadden (1984) present a test for the independence of irrelevant alternatives (IIA) property. They also

¹Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
show that the IIA property can be tested by estimating the model based on a subset of alternatives or by using an alternative model (e.g. a nested logit) and comparing with the MNL models using likelihood ratio, Wald or Lagrange multiplier tests. The likelihood ratio test is easy to perform and is hence often used. McFadden and Train (2000) propose a Lagrange multiplier test that allows to test whether mixing is needed by solely using MNL model estimation results. It is based on the construction of artificial variables and subsequently testing the hypothesis that the artificial variables should be omitted from the MNL model using a Wald or likelihood ratio test. To give yet another example, Fosgerau (2008) describes how the Zheng test (Zheng, 1996) can be applied to discrete choice models. The IM test is complementary to these tests: it is a general test of model misspecification and unlike the other tests it does not require the estimation and comparison of multiple models. The IM test does not give any guidance as to what is the source of misspecification. If the information matrix equality is rejected, the other tests can be used for further investigation, as also suggested by White (1982). In this paper we compare the IM test results with the IIA (Hausman and McFadden, 1984) and McFadden-Train Lagrange multiplier (McFadden and Train, 2000) tests.

Several studies indicate that the finite sample distribution of the IM test when the likelihood function is correctly specified is different from its asymptotic $\chi^2$ approximation. This makes the test prone to reject the null hypothesis when it is true (type I error). A few alternatives to White’s test are proposed in the literature. Chesher (1983) and Lancaster (1984) show how to compute the IM test without the need of third derivatives. Orme (1990) shows that the different variants of the IM test statistic can be improved by including expected values of third order derivatives. Horowitz (1994) also notes that the bootstrap technique is computationally expensive but proposes a cheaper variant to obtain finite sample critical values and compares those with the asymptotic critical values. We briefly explore his approach in our experiments.

The paper is structured as follows. Section 7.2 presents a brief background on maximum likelihood estimation and the IM test. In Section 7.3 we give an overview of MNL route choice models and present the Hessian of the corresponding log-likelihood functions. The Monte Carlo experiment is presented in Section 7.4 and the results based on real data in Section 7.5. In Section 7.6 we discuss the computational time for the different models and tests. Finally, Section 7.7 concludes.
7.2 Maximum Likelihood Estimation and Information Matrix Test

In the context of maximum likelihood estimation (MLE), we aim to solve the following maximization problem

$$\max_\beta \hat{L}_N(\beta) = \frac{1}{N} \sum_{n=1}^{N} \ln f(y_n|\beta)$$

(7.1)

where $f(Y|\beta)$ is some probability density function (pdf), defined on $Y$, conditioned on a set of parameters $\beta$, and $y_1, \ldots, y_N$ are observations. Using the terminology popular in stochastic programming (SP), (7.1) can be seen as the sample average approximation of the “true” problem

$$\max_\beta LL(\beta) = E_Y[\ln f(y|\beta)].$$

(7.2)

We note here that $f$ does not necessarily correspond to the density of $Y$ over the population, in which case the model is said to be misspecified. Assuming that (7.1) and (7.2) have unique solutions, denoted by $\hat{\beta}_N^*$ and $\beta^*$ respectively, we have

$$\sqrt{N}(\hat{\beta}_N^* - \beta^*) \Rightarrow \mathcal{N}(0, H(\beta^*^{-1} I(\beta^*) H(\beta^*)^{-1}),$$

where $\Rightarrow$ denotes the convergence in distribution, $\mathcal{N}$ refers to the normal distribution, $H(\beta^*) = E_Y[\nabla_\beta f(Y|\beta^*)]$, and $I(\beta^*) = E_Y[\nabla_\beta f(Y|\beta^*) \nabla_\beta f(Y|\beta^*)^T]$ is the outer product of scores, also called the Fisher information matrix (see e.g. Newey and McFadden, 1994). In practice, these quantities can be consistently estimated by using their corresponding sample estimators $I_N(\hat{\beta}_N^*)$ and $H_N(\hat{\beta}_N^*)).

The well-known information matrix equality implies that if the model is well specified, i.e. $f(Y|\beta^*)$ is the density of $Y$ over the population, the Fisher information matrix is equal to the opposite of the Hessian matrix,

$$I(\beta^*) = -H(\beta^*).$$

(7.3)

The information matrix equality test proposed by White (1982) is based on the jointly asymptotically distributed Normal property of $D_N(\hat{\beta}_N^*) = I_N(\hat{\beta}_N^*) + H_N(\hat{\beta}_N^*)$

$$\sqrt{N}D_N^\eta(\hat{\beta}_N^*) \Rightarrow \mathcal{N}(0, V_N(\hat{\beta}_N^*)).$$

Here we note that for a matrix $B$, vector $B^\eta$ is defined by taking $\eta$ indicators of interest in $B$. An asymptotic $\chi^2$ statistic test is

$$\varphi_N = ND_N^\eta(\hat{\beta}_N^*) V_N(\hat{\beta}_N^*)^{-1} D_N^\eta(\hat{\beta}_N^*) \Rightarrow \chi^2_{\eta}$$

(7.4)
where \( \chi^2_\eta \) is chi-square distribution with \( \eta \) degrees of freedom. The value of \( D^\eta_N(\hat{\beta}_N^\ast) \) and \( V_N(\hat{\beta}_N^\ast) \) are defined by

\[
D^\eta_N(\hat{\beta}_N^\ast) = \frac{1}{N} \sum_{n=1}^{N} d^\eta_n(y_n|\hat{\beta}_N^\ast) \\
V_N(\hat{\beta}_N^\ast) = \frac{1}{N} \sum_{n=1}^{N} \left[ \psi_n(\hat{\beta}_N^\ast)\psi_n(\hat{\beta}_N^\ast)^T \right]
\]  

(7.5)

where

\[
d_n(y_n|\hat{\beta}_N^\ast) = [\nabla_\beta \ln f(y_n|\hat{\beta}_N^\ast)][\nabla_\beta \ln f(y_n|\hat{\beta}_N^\ast)]^T + \nabla^2_{\beta\beta} (\ln f(y_n|\hat{\beta}_N^\ast))
\]

and

\[
\psi_n(\hat{\beta}_N^\ast) = d^\eta_n(y_n|\hat{\beta}_N^\ast) - \nabla_\beta D^\eta_N(\hat{\beta}_N^\ast)H_N(\hat{\beta}_N^\ast)^{-1}\nabla_\beta \ln f(y_n|\hat{\beta}_N^\ast).
\]

Note that (7.4) contains third derivatives through \( \nabla_\beta D^\eta_N(\hat{\beta}_N^\ast) \) which raises computational concerns. In the following section we present MNL route choice models and the corresponding analytical Hessians so that we can evaluate (7.4) by computing third derivatives by finite difference on the Hessian. We limit the scope to models that can be consistently estimated and that have been used in the literature for analyzing path choices in real uni-modal networks.

### 7.3 Route Choice Models

Consider a transport network represented by a directed connected graph (not assumed acyclic) \( G = (V,A) \) where \( V \) if the set of nodes (vertices) and \( A \) is the set of links (arcs). We denote links by \( k,a \in A \) and the set of outgoing links from the sink node of \( k \) by \( A(k) \). A path \( \sigma \) in this network is a sequence of links \((k_0,\ldots,k_K)\) with \( k_{i+1} \in A(k_i) \) for all \( i < K \).

There are two main issues associated with using discrete choice models to model the probability of choosing paths in \( G \). First, the set of all paths connecting each origin-destination pair, also known as the universal choice set, is unknown and cannot be enumerated. Second, path utilities can be correlated due to, for example, spatial overlap. Existing route choice models can be broadly categorized with respect to how they address these two issues. The MNL model is convenient to address the first issue because it can be consistently estimated on sampled choice sets of paths, correlation can however only be modelled through the deterministic utility. We first give a brief presentation of path-based MNL models, followed by the recursive logit model that is link-based and therefore does not require any choice sets of paths.
7.3.1 Path-based MNL Models

The probability that individual $n$ chooses path $\sigma$ given a sampled choice set of paths $C_n$ is

$$P(\sigma|C_n) = \frac{\prod_{k=1}^{N} e^{x_{\sigma,n} + \delta_c \ln \pi(C_n|\sigma)}}{\sum_{j \in C_n} \prod_{k=1}^{N} e^{x_{j,n} + \delta_c \ln \pi(C_n|j)}}$$  \hspace{1cm} (7.6)

(McFadden, 1978) where $v_{\sigma,n} = v(x_{\sigma,n}, \beta)$ is the deterministic utility component associated with each path $\sigma \in C_n$, $x_{\sigma,n}$ is a vector of observed attributes and $\beta$ is a vector of unknown parameters to be estimated. $\ln \pi(C_n|\sigma)$ is a correction for sampling bias where $\pi(C_n|\sigma)$ is the probability of sampling choice set $C_n$ given that $\sigma$ is the chosen alternative. In order to clarify the difference between state-of-the-art models, we multiply this correction by a binary variable $\delta_c$ that equals zero if utilities are not corrected.

The choice sets $C_n$ can be sampled using some algorithm performing a repeated shortest path search. This corresponds to importance sampling from the universal choice set. In most studies (see e.g. Prato, 2009, for an overview), the sampled choice sets are treated as actual ones and path utilities are not corrected for the sampling ($\delta_c = 0$). In other words, the probability of choosing any other path in the universal choice set is zero. This is a strong, restrictive and in most cases invalid assumption. Frejinger et al. (2009) derive a correction $\pi(C_n|\sigma) = \frac{r_{\sigma,n}}{q(\sigma)}$ where $r_{\sigma,n}$ is the number of times path $\sigma$ was drawn and $q(\sigma)$ is the sampling probability of path $\sigma$. In some applications the correction can be ignored by letting alternative specific constants absorb the sampling bias. This is however not the case in a route choice context because the number of alternative specific constants is too large ($\sum_{n=1}^{N} |C_n|$ where $N$ is the number of observations).

In this paper we refer to model (7.6) with sampling correction ($\delta_c = 1$) as path logit (PL).

The MNL model exhibits the well known IIA property which is expected not to hold in a route choice context, namely, due to spatial overlap among paths. The MNL model is still used, mainly because of the choice set definition issue discussed previously. A simple way to relax the IIA property is to include an attribute in $v_{\sigma,n} \forall \sigma \in C_n$ that depends on other alternatives than $\sigma$ in $C_n$. Several such attributes have been proposed in the literature, for example, commonality factor (Cascetta et al., 1996) and path size (Ben-Akiva and Bierlaire, 1999a). We adopt the one proposed by Frejinger et al. (2009), called expanded path size (EPS), because it takes the path sampling probabilities into account. It is defined as

$$EPS_{\sigma,n} = \sum_{i=0}^{K_{\sigma}} \frac{L(k_i)}{L(\sigma)} \sum_{j \in C_n} \delta(j, k_i) \phi_{j,n}^{\sigma}$$

where $L(\cdot)$ is the length of link $k_i$ or path $\sigma$ and $\delta(j, k_i)$ equals one if $k_i$ is on path $j$ and zero otherwise so that $\sum_{j \in C_n} \delta(j, k_i)$ is the number of paths in $C_n$ that use link $k_i$. $\phi_{j,n}^{\sigma} = \ldots$
max\{1, \frac{1-\|j=\sigma\|}{q(j)\|C_n\|}\} is the expansion factor where \(\|\cdot\|\) is an indicator function and \(\|C_n\|\) is cardinality of \(C_n\). We call the PL model with the EPS attribute path size logit (PSL).

The estimation of PL and PSL corresponds to a quasi-maximum likelihood estimation problem, for which consistency has been established (McFadden, 1978). It is however known that even for consistent estimators, the information identity may be violated (Newey and McFadden, 1994, White, 1982). To the best of our knowledge, there is currently no proof that the information matrix equality holds for correctly specified MNL models when the alternatives are sampled. We therefore provide a proof in C.1.

In order to compute the test statistic in (7.4) we need the analytical Hessian of the log-likelihood function. It is easy to show that for PL/PSL and linear-in-parameters utility functions it is

\[
\nabla^2_{\beta \beta} \hat{L}_N(\beta) = \frac{1}{N} \sum_{n=1}^{N} \psi_n(\beta)\psi_n(\beta)^T - \Omega_n(\beta)\varphi_n(\beta) \quad \text{(7.7)}
\]

where \(\psi_n(\beta) = \sum_{j \in C_n} \nabla_\beta v_{jn}(\beta)e^{v_{jn}(\beta)}\), \(\Omega_n(\beta) = \sum_{j \in C_n} (\nabla_\beta v_{jn}(\beta))^T e^{v_{jn}(\beta)}\) and \(\varphi_n(\beta) = \sum_{j \in C_n} e^{v_{jn}(\beta)}\).

### 7.3.2 Recursive Logit

Fosgerau et al. (2013a) propose the RL model that allows to model the choice of path as a sequence of link choices in a network without defining any choice sets of paths or imposing restrictions on the network. The model is defined in a dynamic discrete choice framework using the same assumptions as Rust (1987).

An instantaneous utility \(u(a|k; \beta) = v(a|k; \beta) + \varepsilon(a)\) is associated with each outgoing link (aka action) \(a \in A(k)\) from a given state/link \(k\). The deterministic utility \(v(a|k; \beta)\) is assumed negative for all links except for a dummy link \(d\) that is zero. \(d\) is added to the network to define the destination as an absorbing state. The random terms are i.i.d. extreme value type I so that the choice model at each decision stage is MNL,

\[
P^d(a|k) = \frac{e^{\frac{1}{\theta}(v(a|k)+V^d(a))}}{\sum_{a' \in A(k)} e^{\frac{1}{\theta}(v(a'|k)+V^d(a'))}} = e^{\frac{1}{\theta}(v(a|k)+V^d(a)-V^d(k))},
\]
and the probability of a path observation is consequently

\[ P^d(\sigma, \beta) = \prod_{i=0}^{K-1} P^d(k_{i+1}|k_i) \]

\[ = \prod_{i=0}^{K-1} e^{\frac{1}{\mu}[v(k_{i+1}|k_i) + V^d(k_{i+1}) - V^d(k_i)]] \]

\[ = e^{-\frac{1}{\mu}V^d(k_0)} \prod_{i=0}^{K-1} e^{\frac{1}{\mu}v(k_{i+1}|k_i)}. \]

The value function \( V^d(a) \) is the expected maximum utility from the sink node of \( a \) to the destination and they can be efficiently computed, for all states and a given destination, by solving the following system of linear equations

\[ [I - M^d(\beta)]z^d(\beta) = b. \quad (7.8) \]

\( z^d(\beta) \) is a vector with elements \( z^d_k = e^{\frac{1}{\mu}V^d(k)} \), \( b \) is a vector of size \([(|A|+1) \times 1]\) with zero values for all states except for the destination that equals one, \( b_d = 1 \) and \( I \) is the identity matrix. Moreover the matrix \( M^d \) is defined by

\[ M^d_{ka}(\beta) = \begin{cases} 
\delta(a|k)e^{\frac{1}{\mu}v(a|k)} & \forall k \in A \\
0 & k = d
\end{cases} \]

where \( \delta(a|k) = 1 \) if \( a \in A(k) \) and 0 otherwise. Similar to the PSL model, it is possible to relax the IIA property by including an attribute in the link specific deterministic utilities that depends on the network characteristics. Fosgerau et al. (2013a) proposed the link size (LS) attribute for this purpose. It is simply the normalized flow given by a RL model with some chosen parameter values \( \tilde{\beta} \) (not the same as the one to be estimated, \( \beta \)) when the demand at the origin is fixed to one

\[ LS^{od} = F^{od}(\tilde{\beta}). \quad (7.9) \]

We note that even though the RL model is based on the same assumptions as Rust (1987), there are some important differences. We assume deterministic attributes in the network so that the next state is deterministically given by the action. The transition probabilities are hence degenerate which considerably simplifies the model. The computational time associated with estimating the model mainly stems from the fact that the value functions are destination specific, or even origin-destination specific if a LS attribute (7.9) is included in the instantaneous utilities. This means that in order to evaluate the log-likelihood function, we need to solve as many systems of linear equations (7.8) as there are destinations or origin-destination pairs in the observations.
The choice probabilities given by the RL model are based on the universal choice set, hence, there is no sampling correction. The information matrix equality can be established similarly to the general maximum likelihood estimation problem (see for instance Amemiya, 1985, Chapter 1).

It is straightforward to derive the Hessian for the RL model, although its expression has a complicated form. The log-likelihood function is

$$\hat{LL}_N(\beta) = \frac{1}{N} \sum_{n=1}^{N} \ln P^{d_n}(\sigma_n, \beta) = \frac{1}{\mu N} \sum_{n=1}^{N} \left( v(\sigma_n, \beta) - V^{d_n}(k_0^n, \beta) \right) \quad (7.10)$$

where $v(\sigma_n, \beta) = \sum_{i=0}^{K_n-1} v(k_{i+1}^n|k_i^n)$ is the sum of the deterministic link utilities of observed path $\sigma_n$, $d_n$ is a destination that corresponds to path observation $\sigma_n$, and its Hessian is

$$\nabla^2_{\beta\beta}\hat{LL}_N(\beta) = -\frac{1}{\mu N} \sum_{n=1}^{N} \nabla_{\beta\beta} V^{d_n}(k_0^n, \beta)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left( \frac{\nabla^2_{\beta\beta} z^{d_n}(k_0^n, \beta)}{z^{d_n}(k_0^n, \beta)^2} \right) + \left( \frac{\nabla_{\beta} z^{d_n}(k_0^n, \beta) \nabla_{\beta} z^{d_n}(k_0^n, \beta)^T}{z^{d_n}(k_0^n, \beta)^2} \right). \quad (7.11)$$

This equation is a specific case of the formulas presented in Eberwein and Ham (2008a). Compared to the general case we benefit from the fact that the derivatives of the value functions can be computed by solving systems of linear equations (details are given in C.2).

### 7.4 Monte Carlo Experiment

In order to assess the size and power of the IM test for different models and sample sizes, we perform a Monte Carlo experiment. We also compare the performance of the IM test with the IIA (Hausman and McFadden, 1984) and the McFadden and Train’s (McTr) Lagrange multiplier (McFadden and Train, 2000) tests. The latter requires to sum over all alternatives in the choice set to compute artificial variables. This makes the test impractical for the RL model since it is a logit over all paths in the universal choice set (cannot be enumerated). Consequently, we only apply the McTr test on the PL model.

We simulate sets of observations with a chosen utility specification of the PL, RL and RL-LS models. In this case the true model is known and we can evaluate the number of type I errors at a given significance level. In order to assess the power of the test we need to estimate and test misspecified models. In this case we simulate observations using the RL-LS model. Based on these observations we estimate and test the RL and PL models and evaluate the number of type II errors. In the following we present the setup of the Monte Carlo experiment in more
detail, explaining how we (i) simulate observations, (ii) sample choice sets of paths for the PL model and (iii) compute the IIA and McTr test statistics.

7.4.1 Experimental Setup

We use the Borlänge network in Sweden that is composed of 3077 nodes and 7450 links. A set of 1832 real path observations is available for this network, corresponding to 1420 different OD pairs and 466 destinations. For more details we refer the reader to Frejinger and Bierlaire (2007) or Fosgerau et al. (2013a) that are based on the same data.

For each model, we simulate 100 sets of observations of size 1832, 500 and 100, hence resulting in a total of 300 sets of observations per model and a total of 243200 simulated paths. The OD pairs are chosen based on the real observations: for the size 1832 we use the OD pair of each real observation. For the smaller sample sizes we randomly draw, with replacement, 500 or 100 OD pairs among the 1420 OD pairs, according to a uniform distribution. For each of the drawn OD pairs, we simulate one observation.

For the purpose of this experiment we can only use models for which we can simulate observations based on the universal choice set. The RL and RL-LS models are fast and easy to apply. The path-based models are more problematic since we cannot enumerate the universal choice sets of paths. The RL and PL models are equivalent so we can use RL to simulate observations also for PL. However, the universal choice set must be enumerated in order to simulate observations with the PSL model, we therefore exclude it from the Monte Carlo experiment.

We define the same deterministic utilities for all three models. We use the following network attributes: $TT(a)$ travel time in minutes of link $a$, left turn $LT(a|k)$ (angle larger than 40 degrees and less than 177 degrees going from $k$ to $a$), u-turn $UT(a|k)$ (angle larger than 177 degrees) dummies and a link constant $LC(a)$ that equals one for all link $a \in A$. The deterministic utility for a link $a$ given a state $k$ or path $\sigma_n = (k_0^n, \ldots, k_{K_n}^n)$ for each model are

$$v_{RL}(a|k) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k)$$

$$v_{RL-LS}(a|k) = \beta_{TT}TT(a) + \beta_{LT}LT(a|k) + \beta_{LC}LC(a) + \beta_{UT}UT(a|k) + \beta_{LS}LS(a)$$

$$v_{PL}(\sigma_n) = \mu[\beta_{LL}TT(\sigma_i) + \beta_{LT}LT(\sigma_i) + \beta_{LC}LC(\sigma_i) + \beta_{UT}UT(\sigma_i)] + \ln \left( \frac{r_{\sigma_n}}{q(\sigma_n)} \right)$$

where $TT(\sigma_n) = \sum_{i=0}^{K_n} TT(k_i^\sigma)$, $LT(\sigma_n) = \sum_{i=0}^{K_n-1} LT(k_{i+1}^\sigma|k_i^\sigma)$, $LC(\sigma_n) = \sum_{i=0}^{K_n} LC(k_i^\sigma)$ and $UT(\sigma_n) = \sum_{i=0}^{K_n-1} UT(k_{i+1}^\sigma|k_i^\sigma)$. The term $\ln(r_{\sigma_n}/q(\sigma_n))$ is the sampling correction in (7.6).
The chosen parameters values are $\tilde{\beta}_\text{obs}^{\text{TT}} = -2.0$, $\tilde{\beta}_\text{obs}^{\text{LT}} = -1.0$, $\tilde{\beta}_\text{obs}^{\text{LC}} = -1.0$, $\tilde{\beta}_\text{obs}^{\text{UT}} = -20.0$ and $\tilde{\beta}_\text{obs}^{\text{LS}} = -1.0$. These values are the same as the ones used in Fosgerau et al. (2013a). Note that we exclude u-turns by fixing the corresponding parameter to a large negative value.

The PL model requires sampled path choice sets $C_n$. In order to correct the utilities the path sampling probability $q(\sigma_n)$ must be known. This severely limits the choice of path generation algorithm. In our case we use the RL model because it allows to efficiently sample paths and compute the corresponding sampling probabilities. We use the deterministic utilities $v^{\text{RL}}(a|k)$ with parameters $\tilde{\beta}_\text{sampl}^{\text{TT}} = -1.8$, $\tilde{\beta}_\text{sampl}^{\text{LT}} = -0.9$, $\tilde{\beta}_\text{sampl}^{\text{LC}} = -0.8$ and $\tilde{\beta}_\text{sampl}^{\text{UT}} = -4$) and make 50 draws for each observation. We note that these values are different from the ones used to simulate observations. In this case, the magnitude of the parameter values is smaller to ensure that diverse paths are sampled. The box plot in Figure 7.1 shows the number of choice sets having a given size for $|C_n|, n = 1, \ldots, 1832$ and over the 100 samples. We note that certain origins and destinations are close which explains the share of choice sets with few paths. Since we draw paths with replacement, some are drawn several times and this is why there is no choice set with 50 paths. Finally we note that we do not investigate the influence of the number of draws, we have simply verified that 50 draws is enough to obtain consistent parameter estimates for the application at hand.

![Box plot](image)

**Figure 7.1:** Box plot of number of paths in $C_n$ for $n = 1, \ldots, 1832$ and over 100 samples

In this experiment we also compare the results of the IM test with the IIA and McTr tests. The IIA test statistic is based on the parameter estimates and covariance matrices of two models. One is estimated on the universal choice set and the other on a subset of the alternatives and hence also a subset of the observations. In our context, the issue lies in defining a subset of alternatives...
for the recursive logit model. We do so by modifying the link-to-link incidence matrix. More precisely, we randomly remove 1000 connections out of the 21,452 non-zero elements in the original incidence matrix. Since the removal of certain connections may result in a graph that is not connected, we remove the connections in a trial and error manner ensuring that the graphs remains connected. The resulting graph contains fewer connections but the same number of links and nodes as $G$. Since a share of observations use the removed connections, the samples do no longer have the sizes 1832, 500 and 100 but are in the intervals $[876, 909]$, $[239, 265]$, $[28, 42]$, respectively.

We apply the McTr test by computing artificial variables based on sampled choice sets and the travel time attribute. We then use the likelihood ratio test to compare the two models, that is, test the hypothesis that the artificial variables should be omitted from the model.

### 7.4.2 Numerical Results

We start by analyzing the type I errors. Table 7.1 reports the number of times we reject the null hypothesis, at the 0.05 significance level, over the 100 samples of observations. The first three columns concern the IM test, the following three the IIA test and the last one the McTr test (only for the PL model as we explained in the beginning of this section). If we had a large number of samples, we would expect the same share of rejections as the significance level. For computational reasons we have however limited this study to 100 repetitions. The results show that the IM test performs well for the larger samples but the size of the test statistic is poor for small samples. For the PL and RL models there are no type I errors for the sample size 1832, while for smaller sample sizes (100 and 500) there are several type I errors.

We note that the performance of the test depends on the model. Indeed, the share of type I errors is larger for RL-LS than the other two models. For the size of 1832 observations, 7 out of 100 models are falsely rejected. As expected, the performance improves with the size of the samples. We play with the sample size of RL-LS only and increase it to 10 times the real one (simulating 100 samples and ten observations instead of one for each real observation) in which case there are 3 type I errors for the IM test and 4 type II errors for the IIA. In this experiment, the IIA test performs slightly worse than the IM test, in particular for the sample size of 500 and 100. Interestingly, the McTr test performs better than both the IM and IIA tests on the smallest samples. Based on these results, sample sizes of only 100 and 500 observations are clearly not enough for both the IM and IIA tests. We report the cumulative distributions of $p$-values in C.3.

We now turn our attention to the power of the tests. Recall that the observations have been simulated using the RL-LS model. This model relaxes the IIA property by including a network attribute in the utility so that the utility of a given path depends on other paths in the network.
We test the RL and PL models which in this context are misspecified. The only source of misspecification is that the IIA property is relaxed, which may also be viewed as an omitted variable.

Horowitz (1994) considers using size-corrected critical values instead of the asymptotic critical values for the IM test. The advocated procedure is similar to the bootstrap technique, except that the parameters are kept to their initial estimates instead of being evaluated for each re-sampled population. Horowitz shows that, under some conditions, the convergence order of the size-corrected critical is higher than the asymptotic critical values, while the computational cost remains reasonable. In our experiments, we use 200 Monte Carlo repetitions to sample from the null hypothesis model with pseudo-true parameter values to obtain the empirical distribution of the IM test statistic. At the 0.05 significance level, the size-corrected critical value is the 0.95 quantile of this distribution.

In Table 7.2 we report the number of type II errors (null hypothesis is not rejected) at the 0.05 significance level. The distributions of the $p$-values are reported in C.3. The power of both the IM and IIA tests is strong for the sample size of 1832 since the null hypothesis is rejected for all models. For samples with only 100 observations, the IIA test has better power than the IM test that rejects just a few misspecified models. For samples with 500 observations, the IM test performs better than the IIA test for the RL model, but worse for the PL model. We note that the power of the McTr test is poor even for samples of 500 observations. The results for the IM test based on the size-corrected critical values are reported in parentheses. For samples with 100 observations, the number of type II errors using size-corrected values is larger than the number given by the asymptotic critical values. While these results seem in contradiction with the conclusions in Horowitz (1994), we note that Horowitz only establishes a better convergence order for the critical values. Therefore, due to the hidden constant, there is no guarantee that for a very small population, the critical values are closer to their nominal values. When the sample size is larger (500 observations), the number of type II errors decreases when using size-corrected critical values, as expected.

In this experiment we have used the conventional significance level of 0.05. Given the small number of repetitions (only 100) we do not expect to have the same share of rejections as the significance level. If we increase the significance level, the number of type I errors increases and

<table>
<thead>
<tr>
<th>Number of observations</th>
<th>IM test</th>
<th>IIA test</th>
<th>McTr test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>RL-LS</td>
<td>PL</td>
</tr>
<tr>
<td>18320</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>1832</td>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>500</td>
<td>6</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>17</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 7.1: Number of type I errors, 100 repetitions, 0.05 significance level
the number of type II errors decreases. For example, we consider the RL model and the 0.2 significance level, the number of type I errors are 5, 18, 43 and the number of type II are 0, 0, 62 for the sample sizes of 1832, 500 and 100, respectively.

Before discussing the numerical results based on the real data, we make a final remark on the PL model without sampling correction (Equation (7.6) with \( \delta = 0 \)). If the universal choice set contains all paths in the network, the information matrix equality only holds for the PL model when the utilities are corrected for the path sampling (see proof in C.1). Numerous studies in the literature report estimation results using sampled choice sets of paths but without sampling correction. We applied the IM test for this case (1832 observations) and all models are strongly rejected with \( p \)-values very close to zero (less than \( 10^{-5} \)).

### 7.5 Numerical Results using Real Data

In this section we report results based on the 1832 real observations. We use the same deterministic utilities as in the previous section. We also test the PSL model that we did not include in the Monte Carlo experiment and the corresponding deterministic utilities are

\[
v_{PSL}(\sigma_n) = \mu[\beta_{LL}TT(\sigma_n) + \beta_{LT}LT(\sigma_n) + \beta_{LC}LC(\sigma_n) + \beta_{PS}EPS(\sigma_n) + \beta_{UT}UT(\sigma_n)]\ln \left( \frac{r_{\sigma_n}}{q(\sigma_n)} \right).
\]

The estimation results are reported in Table 7.3. The only difference in the RL and RL-LS specifications compared to Fosgerau et al. (2013a) is the estimation of the \( \beta_{UT} \) parameter. While this significantly improves the model fit (there are some u-turns in the observations), the signs and magnitudes of the other parameters remain the same. In order to estimate the PL and PSL models, we sample choice sets of paths in the same way as in the Monte Carlo experiment. The PSL model has a significantly better fit than the PL model. We cannot compare the final log-likelihood values of the path-based and link-based models but we note, as expected, that the parameter estimates of PL and RL are very similar.
Table 7.3: Estimation results for real data

Table 7.4 presents the detailed test results of the IM, IIA and McTr tests for the four models. The null hypothesis of both tests are strongly rejected. Based on the findings in the Monte Carlo experiment we do not expect that this is due to small sample issues. It is interesting to note that the PSL model is popular in the literature because it often leads to better model fit and relaxes the IIA property. However, while the EPS and LS attributes do improve the model fit, they do not affect the outcome of these tests. This implies that other model structures than logit should be investigated.

Table 7.4: Test statistic values for IM, IIA and McTr tests for models estimated on real data

7.6 Remarks on Computational Time

In this section we provide a discussion on computational time since it is the main concern associated with the IM test. It is the reason why we limited the Monte Carlo experiment to 100 samples and also why we did not use the bootstrap as Horowitz (1994).
The computational time stems from (i) maximum likelihood estimation and (ii) the evaluation of the IM, IIA and McTr test statistics. For the estimation of the RL model we use the nested fixed point algorithm (Rust, 1987); the algorithm searching over the parameter space (outer algorithm) is combined with an inner algorithm solving the value functions. In our case, the value functions are simply a solution to a system of linear equations but they are destination (RL model) or even OD (RL-LS model) specific. Recall that the real data consists of 1832 observations, 466 destinations and 1420 OD pairs. The computational times reported in this section for simulated data for the same number of observations.

Our code is implemented in MATLAB and we have used an Intel(R) machine, CPU 1.60GHz, running Fedora version 14. The machine is multi-core processor but we only use one processor per model as the code has not been parallelized. It is important to note that the computational time could be considerably decreased since the evaluation of the log-likelihood function as well as the computation of the IM test statistic could be parallelized. Indeed, it takes less than a second to compute the value functions for a given destination or OD pair, but it takes approximately ten minutes to evaluate the log-likelihood function and its gradient for RL model (466 destinations).

Table 7.5 reports the computational time, in hours, associated with the maximum likelihood estimation and the computation of the test statistics for each model and data set. The non-linear optimization algorithm used to search over the parameter space is a basic trust region method. The model used is a second-order Taylor development of the objective function around the current iterate, built with some approximation of the Hessian. For the sake of comparison, we report the computational time for two different approximations: BHHH (Berndt et al., 1974) and BFGS (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970). The BHHH approximation is based on the information matrix equality and therefore works very well for simulated data while BFGS works better for real data. We note that the computational time for sampling choice sets is included in estimation time for the paths based models (PL and PSL). The computational time for RL-LS is considerably longer than for RL since the value functions are OD and not destination specific.

The computational time associated with the evaluation of the IM test statistics mainly stems from the evaluation of the Hessian. While it is fast for path-based models, Equation (7.7), it is more costly for the RL models, Equation (7.11). The IM test is performed on the full matrix. It means that for a model with $|eta|$ parameters, the number of indicators of interest is $\eta = |\beta|(|\beta| + 1)/2$ which also equals the required number of the third derivatives. Since we compute the third derivatives by finite difference on the Hessian, the number of analytical Hessians that needs to be computed is $|\beta|(|\beta| + 1)/2 + 1$.

It is clear from the results in Table 7.5, that the time associated with the maximum likelihood estimation is negligible compared to the time it takes to compute the IM test statistic. The computational time associated with the IIA test is only about a forth of the IM one for the RL
models. It is however more costly to compute the IIA test than the IM test for the path-based models. We note that the time associated with modifying the network for the IIA test is not taken into account since it was done in a trial and error manner. Finally, the computational times associated with McTr test are negligible compared to the IM and IIA tests. This is due to the fact that the test does not require new sampled choice sets. Indeed, we only need to estimate a new model with additional variables and perform a likelihood ratio test.

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>Nb. of variables</th>
<th>Estimation</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BHHH</td>
<td>BFGS</td>
</tr>
<tr>
<td>Simulated</td>
<td>RL-LS</td>
<td>4</td>
<td>0.96</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>RL</td>
<td>3</td>
<td>0.47</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>PL</td>
<td>3</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>Real</td>
<td>RL-LS</td>
<td>5</td>
<td>2.91</td>
<td>2.42</td>
</tr>
<tr>
<td></td>
<td>RL</td>
<td>4</td>
<td>1.98</td>
<td>1.60</td>
</tr>
<tr>
<td></td>
<td>PSL</td>
<td>4</td>
<td>0.05</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>PL</td>
<td>4</td>
<td>0.03</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 7.5: Computational time, in hours, for estimation and testing

7.7 Conclusions

This paper shows how the IM test can be applied to path- and link-based MNL route choice models. The test statistic contains third derivatives and we address this issue by deriving the analytical Hessian for linear-in-parameters utility functions of the MNL and RL models so that we can compute third derivatives by finite difference.

Several studies in the literature indicate that the finite sample distribution of the IM test when the likelihood function is correctly specified is different from its asymptotic \( \chi^2 \) approximation. This makes the IM test prone to reject the null hypothesis when it is true (type I error). We present a Monte Carlo experiment assessing the size and power of the IM test for three route choice models (RL, RL-LS and PL). Moreover, we compare the IM test results with those of the IIA test (Hausman and McFadden, 1984) and then McTr test (McFadden and Train, 2000). The results confirm that the number of type I errors increases with decreased sample size and indicate that in our case we have issues with sample sizes of 500 or less. We also assess the power of the tests by testing the RL and PL models using observations simulated with the RL-LS model. For the sample size of 1832, the IM and IIA tests correctly reject the null-hypothesis for all models but some errors occur for the McTr test. The results indicate that, as expected, small sample issues occur for the smaller sample sizes (500 and 100 observations), in this case the power of the McTr test is extremely poor. While this Monte Carlo experiment provides some insights on the properties of the tests we emphasize the fact that the number of samples is small and that we have used the same network in all instances.
We also test four route choice models estimated on real observations. The null-hypotheses of the IM and IIA tests are strongly rejected for all four models and the McTr test also rejects all the PL models. It is interesting to note that attributes, such as link-size and path size, that are designed to correct path utilities for correlation, significantly improve model fit but do not influence the outcome of these tests. This implies that alternative model structures, such as other multivariate extreme value models or mixed logit models, should be investigated.

The IM and IIA tests have the advantage over the McTr test that they can be applied to both path- and link-based models. The results show that they have similar performance which may raise a question why one should use the IM test. First, the IM test is a general test for misspecification. As mentioned in White (1982), if the IM test rejects the null hypothesis, other tests can be used to identify the source of misspecification. Second, the IM test does not require the estimation of an alternative model. Although this may not be an issue in many applications, it can actually be tricky for route choice models due to the choice set sampling issues and the fact that the universal choice sets cannot be enumerated. In order to apply the IIA test here we needed to modify the network and verify that it is still connected in a trial and error manner.

Finally, we note that the BHHH approximation (Berndt et al., 1974) is based on the assumption that the information matrix equality holds. If this is not true, the non-linear optimization algorithm using BHHH may fail to converge (Mai et al., 2014).
In this chapter we propose an original approach that allows to quickly estimate a family of MEV models with large choice sets. More precisely, we consider static discrete choice models with network-based correlation structures, and we formulate these models as dynamic discrete choice models on the these networks. We then show that these dynamic models can be estimated quickly using the concept of network flows and the NFXP algorithm (Rust, 1987).

The dynamic discrete choice framework used to formulate MEV models is similar to the one proposed in Chapter 3 (the NRL model). The networks of correlation structures (also known as MEV networks) are however different from transport networks in the route choice context, as they have multiple destinations for each observation and the link utilities are defined based on the structural parameters of the MEV models. In practice, the MEV-networks are typically cycle-free, so the value iteration can converge to the fixed point solution after a few iterations, and using dynamic accuracy as proposed in Chapter 3 is not needed.

The proposed dynamic programming technique can have an impact in any large-scale discrete choice application because it is general and allows to estimate parameters and compute predicted choice probabilities in short computational time.

The paper was presented at the 4th International Choice Modeling Conference, Austin, Texas, U.S.A. A preliminary version of the paper is available as a CIRREL'T technical report (Mai et al., 2015e). The paper has been submitted to Transportation Research Part B for possible publication.
A Dynamic Programming Approach for Quickly Estimating Large-scale MEV Models

Tien Mai¹  Emma Frejinger ¹  Mogens Fosgerau²  Fabian Bastin ¹

Abstract

We propose a way to estimate static Multivariate Extreme Value (MEV) models with large choice sets in short computational time. Following Daly and Bierlaire (2006), the correlation structure is defined by a rooted, directed graph where each node without successor is an alternative. We formulate a family of MEV models as dynamic discrete choice models on the graphs of correlation structures and show that the dynamic models are consistent with MEV theory and generalize the network MEV model (Daly and Bierlaire, 2006). Moreover, we show these models can be estimated quickly using the concept of network flows and the nested fixed point algorithm (Rust, 1987).

We present numerical results based on simulated data with varying number of alternatives and nesting structures. We estimate large models, for example, a cross-nested model with 200 nests and 500,000 alternatives and 210 parameters that needs between 100-200 iterations to converge (4.3 hours on an Intel(R) 3.2GHz machine using a non-parallelized code).

Keywords: multivariate extreme value models, dynamic programming, discrete choice, maximum likelihood estimation, nested fixed point algorithm, value iteration.

8.1 Introduction

Discrete choice models with large choice sets are of interest in various applications such as route and location choice analysis. Large number of alternatives can make the estimation and application of models impractical. In this context, sampling alternatives becomes appealing. McFadden (1978) shows how to obtain consistent estimates with sampled choice sets for the multinomial logit (MNL) model. More recently, Guevara and Ben-Akiva (2013a) extend the work of McFadden (1978) for the MEV model. Their approach has been used with a cross-nested logit route choice model (Lai and Bierlaire, 2015) and a mode destination choice model with simulated data (Daly et al., 2014). Sampling alternatives may however be difficult and time consuming in practice, both for estimating model parameters and for making predictions. In this paper we propose a novel approach based on dynamic discrete choice models to deal with the estimation of large-scale MEV models without sampling of alternatives.

¹Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
²Technical University of Denmark, Denmark, and Royal Institute of Technology, Sweden.
Dynamic discrete choice models are in general more complex to estimate and to apply than static discrete choice models since they require dynamic programming problems to be solved in order to evaluate the log-likelihood function. Recently, Fosgerau et al. (2013a) and Mai et al. (2015c) showed that a dynamic discrete choice formulation of the network path choice problem is actually simpler to deal with than the static, path-based, discrete choice model. This paper builds on a similar idea but in a different context. We propose a dynamic discrete choice approach that allows large-scale Multivariate Extreme Value (MEV) models (McFadden, 1978) to be estimated in short computational time.

Daly and Bierlaire (2006) show how to define choice probability generating functions (CPGF) for a certain kind of MEV models (Fosgerau et al., 2013b). These models are defined by a rooted, cycle-free graph with parameters associated with nodes and arcs. This graph is referred to as an MEV-network (also known as Generalized Extreme Value, GEV, network). This way of defining an MEV model is useful because we can just verify that the graph is an MEV-network, then this is an easy way to prove that the CPGF satisfies the MEV properties. This paper complements and extends the work by Daly and Bierlaire (2006).

Based on ideas from route choice modeling we consider the graph as a transport network where the root is an origin and nodes representing alternatives are viewed as destinations. A unit flow is inserted at the origin and is transported via the network, such that the expected flows arriving at the destinations correspond to probabilities. The transition probabilities of a flow between two nodes are defined based on a dynamic discrete choice model, in a way similar to the route choice model proposed by Mai et al. (2015c). We prove that the dynamic model is equivalent to a network MEV model (Daly and Bierlaire, 2006). Moreover, we prove that even if the graph contains cycles, the dynamic model is consistent with MEV theory.

We seek to estimate large-scale models quickly. The main challenge lies in the definition and the computation of the expected maximum utility from a node in the graph to the nodes representing the alternatives and in the computation of choice probabilities. The expected maximum utility at a node in the graph is known in the context of dynamic discrete choice models as the value function. The value functions are computed using a value iteration method, while the choice probabilities are computed as the expected flows from the root to destinations, which can be found as the solution to a system of linear equations. Moreover, efficient non-linear programming techniques require analytical derivatives of the log-likelihood function with respect to the unknown parameters. We show that also the derivatives of choice probabilities are the solutions to systems of linear equations. We use the nested fixed point algorithm proposed by Rust (1987) to estimate the model by maximum likelihood. We present a discussion on computational complexity and provide some comparisons of computational times that clearly show the superiority of the proposed approach, compared to the standard approach which is based on recursive equations.
In summary, this paper makes two main contributions. First, we formulate a family of MEV static discrete choice models as dynamic discrete choice models on graphs of correlation structures and show that these models, under certain conditions, are consistent with MEV theory and generalize the Daly-Bierlaire network MEV model. Second, we show how these models can be estimated by maximum likelihood in short computational time. The estimation code is implemented in MATLAB and freely available upon request.

The paper is structured as follows. Section 8.2 briefly presents the network MEV model and Section 8.3 presents the dynamic discrete choice approach for formulating static discrete choice models. We provide an illustrative example of a cross-nested logit model in Section 8.4. Furthermore, we show how to estimate the model by maximum likelihood in Section 8.5. Numerical results based on simulated data are presented in Section 8.6. Section 8.7 presents a computational complexity analysis of the dynamic approach and Section 8.8 concludes. In Appendix D.1 and D.2 we provide the proofs of the two theorems presented in the paper. Appendix D.3 derives the first derivative of the log-likelihood, and finally, Appendix D.4 presents the formulas for the elasticities.

8.2 The Network MEV Model

Daly and Bierlaire (2006) propose the network MEV model as a general representation of a class of MEV models based on a network structure. They consider a rooted, connected and cycle-free graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and arcs, respectively. There is at most one arc for any given node pair. Each node without successors defines an alternative $j$ in choice set $C$. The graph defines the correlation structure (and a simple example is the well-known tree of a nested logit model where each leaf is an alternative). A parameter $\mu_k > 0$ is associated with each node $k$ and a parameter $\alpha_{ka} > 0$ with each arc $(k, a)$. The CPGFs are defined as

$$G_k(y) = \begin{cases} y_k^{\mu_k} & \text{if } k \in C \\ \sum_{a \in \mathcal{N}(k)} \alpha_{ka} G_a(y)^{\mu_k/\mu_a} & \text{if } k \in \mathcal{N} \setminus C, \end{cases}$$

(8.1)

where $y_k = e^{U_k}$ ($U_k$ is the deterministic utility associated with the alternative represented by node $k$) and $\mathcal{N}(k)$ the set of successor nodes of node $k$. Daly and Bierlaire (2006) show that if $G$ is cycle-free and $\mu_a \geq \mu_k$, $\forall a \in \mathcal{N}(k)$, then $G_k$, for a given node $k$, is a $\mu_k$-MEV CPGF. That is, the choice probabilities generated by $G_k(y)$ are consistent with random utility maximization theory (McFadden, 1973). Daly and Bierlaire (2006) provide a recursive formula to compute
the choice probability, namely,

\[ P_k(j) = \begin{cases} 
\sum_{a \in \mathcal{N}(k)} P_a(j) \Omega_{ka} & \text{if } k \in \mathcal{N} \setminus C \\
1 & \text{if } k = j \quad \forall j \in C, \\
0 & \text{if } k \in C, k \neq j
\end{cases} \quad (8.2) \]

where \( \Omega_{ka} \) is

\[ \Omega_{ka} = \frac{\alpha_{ka} G_a(y)^{\mu_k/\mu_a}}{G_k(y)}. \quad (8.3) \]

The probability of choosing alternative \( j \) given by \( G_r(y) \) is \( P_r(j) \).

### 8.3 Formulating the Network MEV Model using the Dynamic Discrete Choice Framework

In this section we consider the same definition of the graph \( \mathcal{G} \) as in the previous section, that is, an MEV-network. We focus on formulating choice probabilities using the dynamic discrete choice framework and to explain the derivation we use a transport network analogy. More precisely, we view the root as an origin and the nodes representing alternatives as destinations. There is at least one path connecting the root to any destination. The objective is to define the choice probability of each destination/alternative \( j \in C \). These are obtained by describing the choice problem as a sequential node choice made by a traveler going from the origin (root) to a destination/alternative.

A key idea in our formulation is the fact that the probability of an alternative \( j \in C \) can be defined in terms of incoming flow. We consider a unit outgoing flow from the origin to multiple destinations and assign the choice probability of each alternative as the expected incoming flow at the respective destination. We denote the incoming flow at node \( a \) as \( F(a) \) and the probability of going from \( a \) to \( k \) as \( P(a|k) \), \( a, k \in \mathcal{N} \). The expected incoming flows are

\[ F(a) = D(a) + \sum_{a \in \mathcal{N}} P(a|k) F(k), \quad \forall a \in \mathcal{N}, \quad (8.4) \]

where \( D \) is a vector of size \( |\mathcal{N}| \) with zero values for all the nodes except for the origin which equals one. The choice probabilities \( P(j), j \in C \) are

\[ P(j) = F(j), \forall j \in C. \quad (8.5) \]
It is easy to show that $\sum_{j \in C} P(j) = \sum_{j \in C} F(j) = 1$. Moreover, from (8.4), the probability of choosing $j \in C$ can be written equivalently as

$$P(j) = \sum_{\{k_0, \ldots, k_J\} \in \Phi(j)} \prod_{i=0}^{J-1} P(k_{i+1}|k_i), \quad j \in C,$$

(8.6)

where $\Phi(j)$ is the set of all paths connecting $r$ and $j$. A path is defined by a sequence of nodes $k_0, k_1, \ldots, k_J$ such that $k_{i+1} \in \mathcal{N}(k_i), \forall i = 0, \ldots, J-1$, where $k_0$ is the root $r$ and $k_J$ represents alternative $j$.

Both definitions of $P(j)$ are based on next node probabilities $P(a|k), \forall k, a \in \mathcal{N}$. In the following we present how next node probabilities can be computed using the dynamic discrete choice framework and ideas in Mai et al. (2015c). We associate a utility with each node $a \in \mathcal{N}(k)$ conditional on its predecessor $k$

$$u(a|k; \beta) = v(a|k; \beta) + \frac{1}{\mu_k} (\epsilon(a) - \gamma),$$

where $v(a|k; \beta)$ is a deterministic utility, $\beta$ is a vector of parameters to be estimated, $\mu_k$ is a strictly positive scale parameter associated with node $k$, $\epsilon(a)$ is extreme value type I distributed and i.i.d over $a \in \mathcal{N}(k)$. Euler’s constant $\gamma$ is subtracted to ensure that the random terms have zero mean.

The node probabilities depend on the nodes that are available downstream. This is captured by the expected maximum utility (or value functions) $V(k)$ from a node $k$ to the destinations $j \in C$. We associate a deterministic utility $U_j$ with each alternative $j \in C$ and define $V(j) = U_j$. The model can be considered as a dynamic programming problem with absorbing states $j \in C$, thus the value function $V(k)$ for $k \in \mathcal{N} \setminus C$ is recursively defined by Bellman’s equation

$$V(k; \beta) = \mathbb{E} \left[ \max_{a \in \mathcal{N}(k)} \left\{ v(a|k; \beta) + V(a; \beta) + \frac{1}{\mu_k} (\epsilon(a) - \gamma) \right\} \right], \forall k \in \mathcal{N} \setminus C,$$

or equivalently

$$\mu_k V(k; \beta) = \mathbb{E} \left[ \max_{a \in \mathcal{N}(k)} \left\{ \mu_k \left( v(a|k; \beta) + V(a; \beta) \right) + \epsilon(a) \right\} \right] - \gamma, \forall k \in \mathcal{N} \setminus C,$$

(8.7)

which in this case is the logsum

$$\mu_k V(k; \beta) = \ln \left( \sum_{a \in \mathcal{N}(k)} e^{\mu_k (v(a|k; \beta) + V(a; \beta))} \right), \forall k \in \mathcal{N} \setminus C.$$

(8.8)

In order to simplify the notation we omit from now on $\beta$ from the value functions $V$ and from the node-based utilities $v$. It follows from (8.7) that the probability of node $a$ given node $k$ is
given by the MNL model

\[
P(k|a) = \delta(a|k)e^{\mu_k(V(a)+v(a|k)-V(k))}, \quad \forall k, a \in \mathcal{N},
\]

(8.9)

where \(\delta(a|k)\) equals one if \(a \in \mathcal{N}(k)\) and zero otherwise so that the probability is defined for all \(a, k \in \mathcal{N}\).

We define a vector \(Y\) of size \(|\mathcal{N}|\) with entries

\[
Y_k = e^{\mu_k V_k}, \quad \forall k \in \mathcal{N}.
\]

(8.10)

This allows us to write Bellman’s equation (8.8) as a system of non-linear equations

\[
Y_k = \begin{cases} 
\sum_{a \in \mathcal{N}(k)} e^{\mu_k v(a|k)} \frac{Y_{\mu_a}/\mu_a}{Y_k} & \text{if } k \in \mathcal{N}\setminus \mathcal{C}, \\
e^{U_k} & \text{if } k \in \mathcal{C}.
\end{cases}
\]

(8.11)

Using (8.9), the choice probability of node \(a\) given \(k\) can be written as

\[
P(a|k) = \delta(a|k)e^{\mu_k v(a|k)} \frac{Y_{\mu_a}/\mu_a}{Y_k}, \quad \forall k, a \in \mathcal{N}.
\]

(8.12)

The computation of the choice probabilities using (8.6) and (8.9) requires the vector of value functions \(Y\). The latter can be solved by value iteration as presented in Section 8.5.1 below.

Proposition 8.1 states that if the graph is an MEV-network (Daly and Bierlaire, 2006), then the proposed formulation is equivalent to the network MEV model. Hence, the choice probabilities of special cases of network MEV models, e.g. MNL, nested logit and cross-nested logit, can be computed using the proposed formulation. In this context it is interesting to note that any additive random utility model can be approximated by a cross-nested logit model (Fosgerau et al., 2013b).

**Proposition 8.1.** If the graph \(G\) is a non-empty, cycle-free and \(\mu_k \leq \mu_a, v(a|k) = \frac{1}{\mu_k} \ln \alpha_{ka}, \forall k, a \in \mathcal{N}, a \in \mathcal{N}(k)\), then the resulting model is equivalent to the network MEV model i.e. \(Y_k = G_k, \forall k \in \mathcal{N}\), and the choice probabilities given by (8.2) and (8.4) are identical.

**Proof.** We first prove that \(G_k(y) = Y_k, \forall k \in \mathcal{N}\). Indeed, under the hypotheses of Proposition 8.1 and according to (8.1) we have

\[
Y_k = G_k, \quad \forall k \in \mathcal{C}.
\]

(8.13)
For $k \in \mathcal{N} \backslash C$, from (8.11) and using $\alpha_{ka} = e^{\mu_k v(a|k)}$, 

$$Y_k = \sum_{a \in \mathcal{N}(k)} \alpha_{ka} Y_a^{\mu_k/\mu_a}. \tag{8.14}$$

The result then follows from (8.1), (8.13) and (8.14). Consequently, according to Daly and Bierlaire (2006) $Y_k$ associated with a node $k \in \mathcal{N}$ is a $\mu_k - MEV$ CPGF. Furthermore, from (8.12) we obtain

$$\Omega_{ka} = P(a|k) \quad \forall k \in \mathcal{N} \backslash C, a \in \mathcal{N}(k).$$

This leads to the fact the probabilities given by (8.2) are identical to those given by (8.6).

Theorem 8.2 extends the results of Daly and Bierlaire (2006) by showing that even if the graph contains cycles, the resulting model is consistent with MEV theory.

**Theorem 8.2.** If the graph $\mathcal{G}$ is a non-empty and $\mu_k \leq \mu_a, \forall k, a \in \mathcal{N}, a \in \mathcal{N}(k)$ and Bellman’s equation has a solution, then the model is an additive random utility MEV model with the CPGF $G(e^{U_i}, i \in C) = Y_r$, and $Y_r$ is a $\mu_r - MEV$ CPGF.

We provide the proof of this theorem in Appendix D.1.

### 8.4 Illustrative Example

In this section we take the example of a cross-nested logit model to illustrate that the proposed formulation is equivalent to the CPGF given by Ben-Akiva and Bierlaire (1999a)

$$G(y_1, \ldots, y_J) = \sum_{m \in \mathcal{M}} \left( \sum_{j \in \mathcal{C}} \alpha_{jm} y_j^{\mu_m} \right)^{\mu/\mu_m}, \tag{8.15}$$

where $\mathcal{M}$ is the set of nests, $\mu_m$, $\mu$ and $\alpha_{jm}$ are the parameters of the model and $0 < \mu \leq \mu_m$, $\alpha_{jm} \geq 0, \forall j \in \mathcal{C}, m \in \mathcal{M}$. The corresponding choice probability is

$$P(j) = \sum_{m \in \mathcal{M}} \sum_{m' \in \mathcal{M}} \left( \sum_{j \in \mathcal{C}} \alpha_{jm} e^{\mu_m U_j} \right)^{\mu/\mu_m} \frac{\alpha_{jm} e^{\mu_m U_j}}{\sum_{j' \in \mathcal{C}} \alpha_{jm} e^{\mu_m U_{j'}}, \forall j \in \mathcal{C}. \tag{8.16}$$

In this case the graph consists of three layers: a root $r$, a set of nodes $\mathcal{M}$ representing the nests and a set of destination nodes $C$ representing alternatives (see Figure 8.1).

There is an arc between the root and each nest and arcs from all nests to all alternatives. A parameter $\mu_m$ is associated with nodes $m \in \mathcal{M}$ and $\mu$ with the root ($\mu_r = \mu$). We define $v(a|m) = \frac{1}{\mu_m} \ln(\alpha_{am}) \forall m \in \mathcal{M}, a \in \mathcal{N}(m)$ where $\alpha_{am}$ are nesting parameters but define $v(m|r) =$
0, \forall m \in \mathcal{M}. According to (8.8) and (8.9) the probability of \( a \in C \) for a given \( m \in \mathcal{M} \) is

\[
P(a|m) = \frac{e^{\mu_m(U_a + v(a|m))}}{\sum_{a' \in \mathcal{N}(m)} e^{\mu_m(U_{a'} + v(a'|m))}} = \frac{\alpha_{am}e^{\mu_mU_a}}{\sum_{a \in C} \alpha_{am}e^{\mu_mU_a}},
\]

and the nest probability is

\[
P(m|r) = \frac{e^{\mu_rV(m)}}{\sum_{m' \in \mathcal{M}} e^{\mu_rV(m')}}.\]

Moreover, for each node \( m \in \mathcal{M} \) the value function given by (8.8) is

\[
V(m) = \frac{1}{\mu_m} \ln \left( \sum_{a \in \mathcal{N}(m)} e^{\mu_m(v(a|m)+V(a))} \right).
\]

Using (8.6), (8.17), (8.18) and (8.19), the choice probability of choosing \( a \in C \) is

\[
P(a) = \sum_{m \in \mathcal{M}} P(m|r)P(a|m) = \sum_{m \in \mathcal{M}} \frac{\left( \sum_{a \in C} \alpha_{am}e^{\mu_mU_a}\right)^{\mu_r/\mu_m}}{\sum_{m' \in \mathcal{M}} \left( \sum_{a \in C} \alpha_{am}e^{\mu_mU_a}\right)^{\mu_r/\mu_m}} \frac{\alpha_{am}e^{\mu_mU_a}}{\sum_{a \in C} \alpha_{am}e^{\mu_mU_a}},
\]

which is equivalent to the choice probability given by (8.16).

### 8.5 Maximum Likelihood Estimation

In this section we show how the model can be estimated using the nested fixed point algorithm proposed by Rust (1987). The algorithm combines an outer iterative non-linear optimization algorithm searching over the parameter space with an inner algorithm solving the value functions. The likelihood function and consequently the value functions are evaluated for each iteration of the non-linear optimization algorithm. The speed of the algorithm depends critically on how
fast the value functions can be solved. In the following we present a quick solution method to compute the value functions as well as the choice probabilities and log-likelihood function.

8.5.1 Solving the Value Functions

The value functions are the solution to a system of non-linear equations (8.11). We can solve this system using a simple value iteration method. To do so we introduce matrix notation and define a matrix \( M \) of size \( |\mathcal{N}| \times |\mathcal{N}| \) and a vector \( b \) of size \( |\mathcal{N}| \), with entries

\[
M_{ka} = \delta(a|k)e^{\mu_k u(a|k)}, \quad b_k = \begin{cases} e^{U_k} & \text{if } k \in C \\ 0 & \text{if } k \in \mathcal{N} \setminus C, \end{cases}
\]

and a matrix \( X(Y) \) of size \( |\mathcal{N}| \times |\mathcal{N}| \), with entries \( X(Y)_{ka} = Y_a^{\mu_k/\mu_a}, \forall k, a \in \mathcal{N} \). We can write (8.11) in as

\[
Y_k = \sum_{a \in \mathcal{N}} M_{ka} Y_a^{\mu_k/\mu_a} + b_k, \quad \forall k \in \mathcal{N},
\]

or equivalently

\[
Y = [M \circ X(Y)]e + b,
\]

where \( \circ \) is the element-by-element multiplication and \( e \) is a vector of size \( |\mathcal{N}| \) with value one for all nodes. This equation can be solved by a value iteration. We start with an initial vector \( Y^{(0)} \) and compute a new vector for each iteration \( i \)

\[
Y^{(i+1)} \leftarrow [M \circ X(Y^{(i)})]e + b.
\]

We iterate until a fixed point is found using \( ||Y^{(i+1)} - Y^{(i)}||^2 < \tau \) for a given threshold \( \tau > 0 \) as stopping criteria. It can be shown that if Bellman’s equation has a solution, the value iteration method converges after a finite number of iterations (see for instance Rust, 1987, 1988). If Bellman’s equation has no solution, then the sequence \( \{Y^{(0)}, Y^{(1)} \ldots\} \) is unbounded. Mai et al. (2015c) use value iteration with dynamic accuracy to efficiently compute the vector of the value functions in a real road network which contains cycles. In the case of network MEV models (cycle-free graphs) the value iteration method converges to a unique fixed point after a finite number of iterations (see Theorem 8.3, the proof of this theorem is given in Appendix D.2).

**Theorem 8.3.** If the graph \( \mathcal{G} \) is cycle-free, then there exist an integer number \( K > 0 \) such that \( Y^{(i)} = Y^{(K)}, \forall i \geq K \).

In practice, only a few iterations are needed. For example, two iterations for a MNL model and three iterations for a cross-nested logit model, independently of the number of nodes and the structure of the graph.
8.5.2 Choice Probabilities

We have presented two equivalent definitions of choice probabilities in Section 8.3: one based on all paths connecting the root with the destination/alternative and another based on incoming flow. Enumerating all paths can be expensive while the latter approach is easy to perform, since this just amounts to solving a system of linear equations. We show this in more detail in the following.

We can conveniently write (8.4) in matrix notation as a system of linear equations

\[(I - P^T)F = D,\]  
(8.24)

where \(I\) is the identity matrix and \(P\) is a matrix of size \(|\mathcal{N}| \times |\mathcal{N}|\) with entries \(P_{ka} = P(a|k), \forall k, a \in \mathcal{N}\). The probabilities of the alternatives in \(C\) are just the corresponding entry flows:

\[P(j) = F(j), \forall j \in C.\]  
(8.25)

Bailon and Cominetti (2008) show that \((I - P^T)\) is invertible (Lemma 1), so (8.24) has a unique solution. We can therefore compute the choice probabilities of the alternatives in \(C\) by just solving the system of linear equations in (8.24).

8.5.3 Log-likelihood Function and Non-linear Optimization

We assume that the utilities associated with alternatives, the deterministic utility associated with a pair of nodes \(v(a|k), a \in \mathcal{N}(k)\), and the scales of the model \(\mu_k, k \in \mathcal{N}\) are functions of parameters \(\beta\) to be estimated. They can be a sub-vector of \(\beta\), or constants, or even exponential functions of \(\beta\) as in Mai et al. (2015c). The log-likelihood function is defined over observations \(n = 1, \ldots, N\), is

\[LL(\beta) = \sum_{n=1}^{N} \ln P(i_n|C_n),\]  
(8.26)

where \(i_n\) is the chosen alternative, \(C_n\) the choice set of \(n\) and \(P(i_n|C_n)\) is computed using (8.24).

In large-scale MEV models there can be many parameters associated with nodes \(\mu_k, k \in \mathcal{N}\), and node pair utilities \(v(a|k)\). Not all of them are identifiably from data and normalizations are required. More precisely, as stated by Proposition 8.1 and Theorem 8.2, \(\mu_a \geq \mu_k > 0, \forall a \in \mathcal{N}(k)\), need to be satisfied to ensure the MEV consistency. Since only the ratios of \(\mu_k, \forall k \in \mathcal{N}\), matter in the model, we are free to fix \(\mu_r = 1\). The maximum log-likelihood estimation problem
is therefore formulated as a constrained non-linear optimization problem

\[
\max_{\beta} \quad LL(\beta).
\]

(8.27)

We note that specific models may require additional constraints that are not written here. For instance, in the network MEV model, the expected values of the random terms depend on \(\alpha_{ka}\), \(\forall k, a \in \mathcal{N}\), consequently, may vary over alternatives. It is therefore critical to define normalization constraints on the \(\alpha\) parameters in order to have consistent expected utilities. Daly and Bierlaire (2006) suggest such constraints but they need to be analyzed further. For the cross-nested logit model given by (8.15) and (8.16), the \(\alpha\) parameters could be constrained to sum to one \(\sum_{m \in M} \alpha_{jm} = 1\) (Abbé et al., 2007, Papola, 2004).

Non-linear optimization methods are based on the gradient of the log-likelihood function. We provide analytical derivatives in Appendix D.3 and show that they are solutions to systems of linear equations, which makes computation convenient. Note that in Appendix D.4 we derive the elasticities of the model and show that they also can be obtained by solving systems of linear equations.

8.6 Application to Large Problems

In this section we report the performance of the proposed approach based on simulated data. The objective is to evaluate the computational time required to estimate different MEV models with large choice sets. Simulated data allows us to estimate the true model and we do not need to account for misspecification. This makes of course the non-linear optimization problem easier to deal with, but still constitutes a good basis for comparison. Simulated data also allows us to freely choose the number of alternatives, observations and nesting structure.

For the purpose of this experiment, we use the same attribute values for all observations and we simulate a large number of observations so that the model parameters can be identified. This setup allows us to solve the value functions once each time we evaluate the log-likelihood function. This serves well the purpose of the numerical results since the computational time mainly stems from the evaluation of the value functions. When the value functions are specific to each observation or to different groups of observations, the code associated with the evaluation of the log-likelihood function can be parallelized to decrease the computational time. The most efficient way to do this depends on the application. We provide a more in-depth discussion on computation time in Section 8.7. In the following section we present the experimental setup in more detail.
8.6.1 Experimental Setup

We use three different models (named CNL-1, CNL-2 and N-MEV) to generate observations. The first two models are cross-nested logit models (discussed in Section 8.4) and the corresponding MEV-network is illustrated in Figure 8.2. The two models have different numbers of nests, CNL-1 has 5 and CNL-2 has 200. The third model (N-MEV) is a three-levels cross-nested logit model and the graph of this model is illustrated in Figure 8.3. The first level has 5 nests and the second level has 50 nests, level one nests connect to all level two nests and level two nests connect to all alternatives. Note that the sizes of these graphs depend on the numbers of alternatives in the choice sets. In this experiment we create three choice sets of sizes 10,000, 100,000 and 500,000, denoted by D1, D2 and D3, respectively.

We associate 6 different attributes with each alternative. The attributes are generated uniformly in the interval [0, 5] using independent draws. The deterministic utility of each alternative is specified as a linear-in-parameters formulation of the six attributes. In order to generate observations, for each model we choose manually the values in the interval [−1, −2] for the parameters associated with the alternative utilities. Also, the values of the scales \( \mu_k \) and \( \alpha_{ka} = e^{\mu_{k\{a|k\}}} \), \( \forall k \in \mathcal{N}\setminus C \), \( a \in \mathcal{N}(k) \), are chosen manually in the intervals (0, 0.5) and (0, 1), respectively. For each model we simulate samples of size \( 10^5 \), \( 10^6 \) and \( 2 \times 10^6 \) from choice sets D1, D2 and D3, respectively. A summary is provided in Table 8.1.

<table>
<thead>
<tr>
<th>Models</th>
<th>Nb. alternatives</th>
<th># arcs in graphs</th>
<th># Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNL-1</td>
<td>10,000 (D1)</td>
<td>17,262</td>
<td>100,000</td>
</tr>
<tr>
<td></td>
<td>100,000 (D2)</td>
<td>172,376</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td>500,000 (D3)</td>
<td>862,597</td>
<td>2,000,000</td>
</tr>
<tr>
<td>CNL-2</td>
<td>10,000 (D1)</td>
<td>30,011</td>
<td>100,000</td>
</tr>
<tr>
<td></td>
<td>100,000 (D2)</td>
<td>298,237</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td>500,000 (D3)</td>
<td>1,490,207</td>
<td>2,000,000</td>
</tr>
<tr>
<td>N-MEV</td>
<td>10,000 (D1)</td>
<td>56,972</td>
<td>100,000</td>
</tr>
<tr>
<td></td>
<td>100,000 (D2)</td>
<td>566,751</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td>500,000 (D3)</td>
<td>2,831,668</td>
<td>2,000,000</td>
</tr>
</tbody>
</table>

Table 8.1: Models and simulated data sets

We estimate all the parameters associated with alternative utilities and all the scale parameters \( \mu_k \), \( \forall k \in \mathcal{N}\setminus C \). There are many parameters \( \alpha_{ka} \), \( \forall k \in \mathcal{N}, a \in \mathcal{N}(k) \) and estimating all of them is impractical. For the sake of illustration, for models CNL-1 and CNL-2 we also estimate 4 parameters \( \alpha_{ka} \), \( a \in \mathcal{N}(k) \), associated with two arbitrary alternatives and two nests that these alternatives belong to, the other parameters are fixed at their true values. So model CNL-1 has 10 parameters and 7 constraints, model CNL-2 has 211 parameters and 202 constraints. For the model N-MEV we estimate 56 scale parameters \( \mu_k \), \( k \in \mathcal{N}\setminus C \) and all parameters \( \alpha_{ka} \).
∀ \alpha \in \mathcal{N}(k) \text{ and } k, a \in \mathcal{N}\setminus C. All the remaining parameters are fixed to their true values. In total the N-MEV model has 305 constraints and 312 parameters to be estimated.

The pseudo-code in Algorithm 1 is used to compute the log-likelihood function and its gradient based on the methods proposed in Section 8.5. For maximizing the log-likelihood we use the interior point algorithm with BFGS to solve the constrained optimization problems. In particular, for the large-scale models (CNL-2 and N-MEV) we use the limited memory BFGS algorithm (L-BFGS) (see for instance Nocedal and Wright, 2006, Chapter 9).

Our code is implemented in MATLAB 2015 and we have used an Intel(R) 3.20 GHz machine with a x64-based processor. It is running in the Window 8 64-bit Operating System. The machine has a multi-core processor but we only use one processor to estimate the models as the code is not parallelized.

### 8.6.2 Computational Time Results

In Table 8.2 we report the results in terms of the computation time required to estimate the models for the different data sets. Given that there are many parameters, we do not report the parameter estimates but note that they are not significantly different from their true values according t-tests at the 0.05 significance level.
Algorithm 1: Log-likelihood($\beta$)

begin
\# 1. Compute the value functions
\[ Y = 0; \]
do
\[ Y_{prev} = Y; \]
\[ Y = [M \circ X(Y)].e + b; \]
while $Y \neq Y_{prev}$:
\# 2. Compute the gradients of the value functions, $nb_{\text{pars}}$ is the number of parameters, see Appendix D.3
for $i = 1$ to $nb_{\text{pars}}$
do
\[ \frac{\partial V}{\partial \beta_i} = (I - L)^{-1}(Re + h); \]
end
\# 3. Compute the choice probabilities
\[ F = (I - P^T)^{-1}D; \]
\# 4. Compute the gradients of the choice probabilities, see Appendix D.3
for $i = 1$ to $nb_{\text{pars}}$
do
\[ \frac{\partial F}{\partial \beta_i} = (I - P^T)^{-1}\frac{\partial P^T}{\partial \beta_i}F; \]
end
\# 5. Compute the log-likelihood function and its gradient

For the choice set D1 it takes around 5 seconds to estimate model CNL-1 and less than 2 minutes for estimating CNL-2. For the largest choices set (D3), the estimation times are about 20 minutes for model CNL-1 and around 4 hours for model CNL-2. It is very fast to compute elasticities, 5 seconds for the largest choice set with CNL-2. We note that for all the cross-nested logit models, the optimization algorithm needs around 100 to 300 iterations to converge.

We now turn our attention to the results for model N-MEV which is more time consuming to estimate. The non-linear optimization algorithm requires from 300 to 500 iterations to converge (the algorithm needs less than 50 iterations if the scale parameters $\mu_k$ are not estimated). The results show that we can estimate a large-scale three-levels network MEV model (N-MEV) with a large choice set (D3: 500,000 alternatives) in approximately 14 hours.

Finally, we note that if the alternative utilities are observation specific, then we need to compute the value functions and their gradients for each observation. In this case, the computational time increases proportionally to the number of observations if the code is not parallelized. For example, if the number of observations is 1000, then the computation of the log-likelihood and its gradients would take approximately 140 seconds for CNL-1 with D1, 1750 seconds for CNL-1 with D2, and more than 5 hours for CNL-2 with D2. For large number of observations it is obvious that the log-likelihood function and its gradient should be computed using several processors in parallel.
8.7 Comparisons of Computational Time

Based on the same experimental setting as in the previous section, the objective of this section is to provide a discussion of computational time of the dynamic programming approach as well as to present comparisons with the standard approach. The computations using the dynamic approach can be done as sparse matrix operations while the standard approach relies on recursive operations. In order to make the discussion application oriented, we compare the computational time based on the case that the utilities are observation specific. In this case, the value functions and their gradients need to be solved for each observation.

Algorithm 1 shows the pseudo code to compute the log-likelihood and its gradient using matrix operations. All the matrices in Algorithm 1 have the size of $|\mathcal{N}| \times |\mathcal{N}|$. They are sparse and the number of non-zero elements is $|\mathcal{A}|$, i.e., the number of arcs in the graph. So the complexity of matrix operations is typically proportional to $|\mathcal{A}|$ (Gilbert et al., 1992). We note that, when the value functions are observation specific, given an observation, four steps #1, #2, #3 and #4 need to be performed to compute the choice probability as well as its gradients. In the following we discuss the computational complexity of those steps in more detail.

We denote $\phi(|\mathcal{A}|)$ the computational time for each iteration in the value iteration method and $\varphi(|\mathcal{A}|)$ the computational time for solving a system of linear equations $Ax = b$, where $A$ is a matrix of size $|\mathcal{N}| \times |\mathcal{N}|$ with $|\mathcal{A}|$ non-zero elements. The total computational time to evaluate an observation probability and its gradient using a single processor can be approximated by

$$CT^M = nb_{iters} \times \phi(|\mathcal{A}|) + (2nb_{pars} + 1) \times \varphi(|\mathcal{A}|),$$

where $nb_{iters}$ is the number of iterations required for the value iteration method.

The estimation of the network MEV model can also be carried out using the recursive functions proposed by Daly and Bierlaire (2006). In this case, an observation probability and its gradients

<table>
<thead>
<tr>
<th>Models</th>
<th>Choice sets</th>
<th>Computational time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LL and gradient</td>
</tr>
<tr>
<td>CNL-1</td>
<td>D1</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>D3</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>D5</td>
<td>7.84</td>
</tr>
<tr>
<td>CNL-2</td>
<td>D1</td>
<td>1.44</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>18.29</td>
</tr>
<tr>
<td></td>
<td>D3</td>
<td>88.52</td>
</tr>
<tr>
<td>N-MEV</td>
<td>D1</td>
<td>2.74</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>38.47</td>
</tr>
<tr>
<td></td>
<td>D3</td>
<td>162.52</td>
</tr>
</tbody>
</table>

Table 8.2: Computational time (in seconds)
can be computed based on the first 4 steps in Algorithm 1 but using recursive functions instead of matrix operations. More precisely, the CPGFs and their gradients can be computed using (8.1), and the choice probabilities using (8.2). Note that under the hypotheses of Proposition 8.1 we have $Y(a) = G_a$ and $\Omega_{ka} = P(a|k)$, $\forall k, a \in \mathcal{N}$, $a \in \mathcal{N}(k)$. Algorithm 2 shows a recursive implementation to compute $Y(k)$ (or $G_k$), for a node $k \in \mathcal{N}$. The computational time of a recursive implementation is typically proportional to $|\mathcal{A}|$. We denote it $\tau(|\mathcal{A}|)$. The gradients of the CPGFs, the choice probabilities as well as their gradients can be computed by a similar recursive implementation as in Algorithm 2. Finally, the computational time to compute an observation probability and its gradients using recursive functions can be approximated by

$$CT^R = (nb_{pars} + 1) \times \tau(|\mathcal{A}|) + (nb_{pars} + 1) \times \tau(|\mathcal{A}|),$$

(8.29)

where $(nb_{pars} + 1) \times \tau(|\mathcal{A}|)$ is the time associated with the evaluation of the CPGFs and their gradients, and $(nb_{pars} + 1) \times \tau(|\mathcal{A}|)$ is the time associated with the choice probability and the respective gradients. We note that, for a given alternative $j \in C$, (8.2) needs to be performed in order to compute $P(j|C)$.

**Algorithm 2: getYrecur(k) # Compute the CPGF at node k**

```
begin
forall the $a \in \mathcal{N}(k)$ do
  if $Y(a)$ has not been computed then
    if $a \in C$ then
      $Y(a) = e^{U_a}$;
    else
      $Y(a) = getYrecur(a)$;
  end
return $b_k + \sum_{a \in \mathcal{N}(k)} M_{ka} Y(a) \mu_k / \mu_a$;
```

We now turn our attention to a comparison between the computational times given in (8.28) and (8.29). We start with a discussion of the three values $\tau(|\mathcal{A}|)$, $\phi(|\mathcal{A}|)$ and $\varphi(|\mathcal{A}|)$. When solving sparse systems of linear equations, iterative methods are often used. Many studies have focused on how to speed up large scale sparse matrix operations, especially solving sparse systems of linear equations (e.g. Barrett et al., 1994, Gilbert et al., 1992, Saad, 2003). These methods produce an approximate solution after a finite number of iterations through a matrix-vector product or an abstract linear operator (Saad, 2003). This leads to the fact that $\varphi(|\mathcal{A}|)$ only involves product or linear operators. Since $\phi(|\mathcal{A}|)$ involves raising each element of a matrix by a real exponent, $\phi(|\mathcal{A}|)$ and $\tau(|\mathcal{A}|)$ are expected to be larger than $\varphi(|\mathcal{A}|)$.

We now compare $CT^R$ with $CT^M$. Since $\varphi(|\mathcal{A}|)$ is expected to be smaller than $\phi(|\mathcal{A}|)$ and $\tau(|\mathcal{A}|)$, and $\phi(|\mathcal{A}|)$ and $\tau(|\mathcal{A}|)$ may be not significantly different, we expect, for network MEV models with large number of alternatives, $CT^M$ to be smaller than $CT^R$. We provide a numerical example using model CNL-2 with choice set D3.
In Table 8.3 we report the ranges of \( \hat{\phi}(|A|) \), \( \hat{\phi}(|A|) \), \( \hat{\tau}(|A|) \) (approximated values of \( \phi(|A|) \), \( \varphi(|A|) \), \( \tau(|A|) \) when performing Algorithms 1 and 2 in MATLAB). We note that \( nb_{\text{iter}} = 3 \) in Algorithm 1, and in order to achieve good performance, we use a matrix operation to compute the sum in Algorithm 2. This allows us to avoid using a for-loop in MATLAB which is slower, compared to an equivalent matrix operation. As expected, \( \hat{\phi}(|A|) < \hat{\phi}(|A|) \) and \( \hat{\phi}(|A|) < \hat{\tau}(|A|) \). Moreover, \( \hat{\phi}(|A|) < \hat{\tau}(|A|) \). These results indicate that the matrix operations need less time than the recursive functions to solve systems of sparse linear equations. Consequently, the computational time for evaluating a choice probability and its gradient is also smaller, compared to the recursive approach. This is shown in Table 8.4 where we report the computational times for performing the first 4 steps of Algorithm 1.

There are remarkable differences between matrix operations and recursive functions in Steps #2, #3 and #4. The differences in these steps are due to the good performance of solving systems of linear equations. The differences between Steps #3 and #4 are mainly due to the large number of parameters to be estimated (the model CNL-2 has 211 parameters). If we estimate the model with 1000 observations, in order to evaluate the log-likelihood and its gradients, the Steps #3 and #4 take 31 seconds and 8900 seconds, respectively, if using matrix operations while they are 5975 and \( \approx 10^6 \) seconds, respectively, for the recursive functions. Hence, the matrix operation approach is still over a hundred times faster.

It is important to note that, if the value functions are not observation specific, the computational time using the recursive functions still increases proportionally with the number of observations while this number does not significantly affect the computational time using the matrix operations, as shown in the previous section. Hence, in real applications, if the observations can be aggregated into different groups such that the value functions are group specific, then we can solve the value functions once for each group and reduce the computational cost. This is also an advantage of the dynamic programming approach, compared to the recursive functions.

Finally, we acknowledge that there may exist better implementations of the recursive functions that we are unaware of. Nevertheless, the results clearly show the potential of the dynamic programming approach.

| \( \hat{\phi}(|A|) \) | \( \hat{\phi}(|A|) \) | \( \hat{\tau}(|A|) \) |
|---------------------|---------------------|---------------------|
| (0.28:0.32)         | (0.035:0.045)       | (4.5:5.5)           |

Table 8.3: Computational time comparisons (in seconds)

8.8 Conclusion

In this paper we have introduced a novel approach to construct the network MEV model using the dynamic discrete choice framework. We have shown that under certain conditions the
resulting models is consistent with McFadden’s MEV theory and generalizes the network MEV model. We presented a new estimation method that is convenient to use for estimating MEV models with large choice sets i.e. computing choice probabilities using the expected flows in the graph, estimating the model using the nested fixed point algorithm and computing the model derivatives and elasticities by solving systems of linear equations.

We have presented numerical experiments using simulated data. The results indicate that we are able to quickly estimate the cross-nested and multi-levels network MEV models with large choice sets and large number of observations. We also provided a discussion on the computational time of the dynamic programming approach compared to recursive operations and show why it is expected to be faster. Finally, the estimation code is implemented in MATLAB and is available upon request.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Matrix operations</th>
<th>Recursive functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>1.21</td>
<td>4.55</td>
</tr>
<tr>
<td>#2</td>
<td>9.31</td>
<td>1117.8</td>
</tr>
<tr>
<td>#3</td>
<td>0.031</td>
<td>5.975</td>
</tr>
<tr>
<td>#4</td>
<td>8.90</td>
<td>1258.7</td>
</tr>
</tbody>
</table>

Table 8.4: Computational time (in seconds) comparisons of matrix and recursive operations
Chapter 9

Structured Quasi-Newton Techniques for Maximum Likelihood Estimation

In this chapter we examine structured quasi-Newton techniques, which have been intensively studied for least-squares problems but received limited attention for maximum likelihood. We consider the trust region and line search algorithms (see for instance a review in Section 2.2.2) and focus on switching strategies between different Hessian approximations. We assess the numerical efficiency of the proposed switching methods using mixed logit and logit based route choice models.

We estimate a path-based (Frejinger et al., 2009) and a RRM-based RL model (Chapter 6) to assess the performance of the considered optimization algorithms, but note that these algorithms are also used to estimate other recursive route choice models as well as the MEV models considered in Chapter 8. Moreover, it is important to note that the application of our approach is not restricted to the maximum likelihood, as it can also be used for least-squares problems.

This work has been presented at the 21st International Symposium on Mathematical Programming (ISMP) and the CORS/INFORMS 2015 Joint International Meeting, Montréal. A preliminary version of the paper is also available as a CIRRELT technical report (Mai et al., 2014). The paper will be submitted to Computational Management Science for possible publication.
Structured Quasi-Newton Techniques for Maximum Likelihood Estimation

Tien Mai\(^1\)  Fabian Bastin \(^1\)  Michel Toulouse \(^2\)  Manyuan Tao \(^3\)

Abstract

Maximum likelihood estimation is one of the most popular techniques in econometrics due to its strong theoretical appeal, but can lead to severe numerical issues when the underlying optimization problem is solved. If smooth enough, it however enjoys common properties with other data-fitting methods, especially nonlinear least-squares. Various structured quasi-Newton techniques have been proposed to efficiently solve nonlinear least-squares problems, but maximum likelihood problems have received less attention. We examine in this paper a range of trust region and line search algorithms and focus on the impact that an approximation of the Hessian matrix has on their respective performance. In particular, we examine and develop switching strategies for different Hessian approximations, and assess their numerical efficiency for the estimation of discrete choice models, more precisely mixed logit and logit based route choice models.

Keywords: structured quasi-Newton, trust region, line search, maximum likelihood estimation, Hessian approximation, switching methods, discrete choice, mixed logit, logit route choice.

9.1 Introduction

With the growing interest in “big data”, the need for efficient model estimation techniques becomes crucial. Many important data-fitting problems can be formulated as unconstrained optimization problems, for which the Newton approach is appealing. Trust region and line search techniques ensure, under mild conditions, convergence from an arbitrary starting point. These methods typically rely on a second-order Taylor-development of the objective function, therefore requiring the Hessian of the objective function to be available. However, the numerical cost associated with the Hessian evaluation is usually computationally expensive, and one prefers to construct some approximations of the Hessian, leading to so-called quasi-Newton techniques. The most popular Hessian approximations are the BFGS (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970) and the symmetric rank-1 (SR1) update (Conn et al., 1991). Both of them maintain the symmetry of the matrix and satisfy the secant condition.

\(^1\)Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Canada
\(^2\)Vietnamese-German University, Binh Duong, Vietnam, and CIRRELT, Canada
\(^3\)Xiamen University, Fujian, China
Such approximations are usually guaranteed to converge towards the true Hessian over the iterations of the optimization process (see for instance Nocedal and Wright, 2006, Chapter 6), but in practice, this convergence may be slow, and many iterations may be required before we can benefit from the Newton search direction.

Data-fitting problems however enjoy desirable properties ensuring that the Hessian can be efficiently computed if the model is correctly specified. The prominent examples are the Gauss-Newton approach in the context of nonlinear least-squares, along with its globally convergent variants, the damped Gauss-Newton (Dennis, Jr. and Schnabel, 1996) and Levenberg-Marquardt algorithms (Levenberg, 1944, Marquardt, 1963, Moré, 1978), and the BHHH method (Berndt et al., 1974) for maximum likelihood problems, often viewed as superior to standard quasi-Newton methods (see e.g. Carling and Söderberg, 1998, Jeliazkov and Lloro, 2011). However, in case of misspecification, the numerical performance can be poor close to the optimal solution. Structured quasi-Newton techniques aim to address this pitfall by building an approximate correction applied to Hessian approximations. These methods have been extensively studied for nonlinear least-squares problems, but have received limited consideration in the context of maximum likelihood, while slow convergence has been reported for some advanced discrete choice models such as mixed logit models when using the BHHH method (Bastin et al., 2005). We therefore aim to review structured quasi-Newton techniques for maximum likelihood estimation, especially in the context of discrete choice analysis. In particular, we focus on methods that allow to switch between the BHHH matrix and its corrections during the optimization process, using line search or trust region methods.

The paper is structured as follows. We first briefly review structured quasi-Newton principles in Section 9.2. Section 9.3 provides relevant background on maximum likelihood estimation. Section 9.4 briefly describes the trust region and line search optimization methods, and Section 9.5 introduces different Hessian approximation techniques. We present our switching strategies for these two optimization methods in Section 9.6. Section 9.7 introduces some basic concepts of discrete choice theory, as it constitutes the studied framework for our numerical experiments. Numerical assessments are reported in Section 9.8, and finally, Section 9.9 concludes.

### 9.2 Structured Quasi-Newton Methods

Following the presentation in Gay (1979), we consider data-fitting problems expressed as

\[
\min_{\theta} \phi(\theta) \overset{\text{def}}{=} \sum_{n=1}^{N} \rho_n(r_n(\theta)),
\]  

where \( r_n : \mathbb{R}^p \to \mathcal{R} \), \( \rho_n : \mathcal{R} \to \mathcal{R} \), \( N \geq p \), and \( \mathcal{R} \) is the set of real numbers. Moreover, we assume that \( r_n \) and \( \rho_n \), \( n = 1, \ldots, N \), are twice continuously differentiable.
Two important examples of the form (9.1) are the least-squares and maximum likelihood problems. In the least-squares problem, \( \rho_n(\tau) = \frac{1}{2} \tau^2 \), \( \forall n \), and \( r_n(\theta) = \| f_n(\theta) - y_n \| \), given observations \( y_1, \ldots, y_N \). In the maximum likelihood, \( \rho_n(\tau) = -\ln \tau \), and \( r_n(\theta) \) is a probability density or mass value associated with an observation \( y_n \). Let define the mapping \( R(\theta) : \mathbb{R}^p \rightarrow \mathbb{R}^N \) as

\[
R(\theta) = \begin{pmatrix} r_1(\theta) & r_2(\theta) & \cdots & r_N(\theta) \end{pmatrix}^T,
\]

where \( T \) is the transpose operator. We have

\[
\nabla \phi(\theta) = \sum_{n=1}^{N} \frac{d\rho_i}{dr}(r_n(\theta)) \nabla r_n(\theta),
\]
or, if \( \rho_n = \rho \) for all \( n \),

\[
\nabla \phi(\theta) = J(\theta)^T \nabla R \rho(R(\theta)),
\]

where \( J(\theta) \) is the Jacobian of \( R(\theta) \). Moreover, we then have

\[
\nabla^2 \phi(\theta) = J(\theta)^T \nabla^2 R \rho(R(\theta)) J(\theta) + \sum_{n=1}^{N} \frac{d\rho}{dr}(r_n(\theta)) \nabla^2 r_n(\theta).
\]

The first term in the right hand side of (9.2) is typically cheap to compute, compared to the second term, and if \( J(\theta) \) is of full row-rank, it provides a descent direction. Indeed, using similar arguments to Dennis, Jr. and Schnabel (1996), Chapter 10, we consider the Newton-like step defined as

\[
s = -(J(\theta)^T \nabla^2 R \rho(R(\theta)) J(\theta))^{-1} J(\theta)^T \nabla R \rho(R(\theta)).
\]

If \( \nabla \phi(\theta) \neq 0 \), we have

\[
\nabla \phi(\theta)^T s = -\nabla R \rho(R(\theta))^T J(\theta)(J(\theta)^T \nabla^2 R \rho(R(\theta)) J(\theta))^{-1} J(\theta)^T \nabla R \rho(R(\theta)) < 0.
\]

This suggests the possibility to use this step in combination with the Cauchy step in the trust region framework, as it is sufficient to ensure global convergence, or with a line search strategy, as under mild conditions a well-defined backtracking approach will guarantee convergence (see e.g. Theorem 6.3.3 in Dennis, Jr. and Schnabel, 1996). However, neglecting the second term in (9.2) may lead to poor convergence rate as it can prevent to achieve the full Newton step close to the solution.

The conditions ensuring that

\[
\nabla^2 \phi(\theta^*) = J(\theta^*)^T \nabla^2 R \rho(R(\theta^*)) J(\theta^*)
\]

are well established for nonlinear least-squares and maximum likelihood problems. In essence, the equality (9.3) is achieved if the model is correctly specified, and in this case it is safe to
neglect the second term in (9.2). In the least-squares context, this means that the residuals are zero at the solution (see again Dennis, Jr. and Schnabel, 1996, Chapter 10), and leads to the Gauss-Newton approach. For maximum likelihood problems, the information matrix equality ensures that (9.3) is asymptotically valid when the number of observations grows to infinity, for correctly formulated models (see Newey and McFadden, 1994, for a comprehensive coverage of maximum likelihood properties). Berndt et al. (1974) suggested to use this equality in Newton methods, given that the BHHH method is popular in econometrics (see for instance Train, 2003, Chapter 8). When (9.3) does not hold, the second term in (9.2) cannot be ignored anymore. Since the exact Hessian is usually expensive to obtain while only useful close to the solution in order to achieve fast convergence with the full Newton step, it is cheaper to build an approximation of it, with the aim to capture the full Hessian close to the solution. Such approach is referred to as structured quasi-Newton since it exploits some problem properties, and has been extensively studied for least-squares problems, in the trust region (for instance Dennis Jr et al., 1981, Toint, 1987), and line search (Al-Baali and Fletcher, 1986, Spedicato and Vespucci, 1988, Zhou and Chen, 2010). Maximum likelihood problems have received much less attention, and, to our best knowledge, only trust region techniques have been considered (Bunch, 1987, 1988, 2014).

9.3 Maximum Likelihood Estimation

Maximum likelihood is one the most popular technique in statistics to estimate the parameters of a probabilistic model, given some observations assumed to be the realizations of some random vector. More precisely, consider a random vector \( Y \), and assume we have \( N \) observations independently drawn from this vector. Let assume for now that \( Y \) is continuous. Denote by \( d(Y|\theta) \) the probability density function (pdf) \( Y \), conditioned on a set of parameters \( \theta \). The random distribution would be completely characterized if we know the particular value of \( \theta \), say \( \theta_0 \), corresponding to the population under interest. In the discrete case, we would consider the probability mass function instead of the density. Since the observations are assumed to be independent, the joint density is the product of the individual densities:

\[
d(y_1, y_2, \ldots, y_N | \theta) = \prod_{n=1}^{N} d(y_n | \theta).
\]

However, we are not interested in the observations, that are known, but rather in \( \theta \), so it is convenient to consider a function of \( \theta \) that would follow the value of the joint density, given the observations \( y_1, \ldots, y_N \)

\[
L(\theta | y_1, y_2, \ldots, y_N) = d(y_1, y_2, \ldots, y_N | \theta),
\]
where $L(\theta \mid y_1, y_2, \ldots, y_N)$ is called the likelihood function. Since we do not know $\theta_0$, we will approximate it by computing an estimator $\hat{\theta}_N$ of its value, that can be judged as the most likely value for $\theta$, given our observations. This is done by maximizing the function $L(\theta \mid y_1, \ldots, y_N)$ with respect to $\theta$

$$\max_{\theta \in \Theta} L(\theta \mid y_1, y_2, \ldots, y_N),$$

(9.4)

where we confine the search to the parameter space $\Theta$, and we assume that $\theta_0$ belongs to $\Theta$. We assume furthermore that (9.4) has a unique solution, called the maximum likelihood estimator

$$\hat{\theta}_N = \arg \max_{\theta \in \Theta} L(\theta \mid y_1, y_2, \ldots, y_N).$$

In practice, due to numerical stability issues, it is often more convenient to work with the logarithm of the likelihood function, called the log-likelihood

$$\ln L(\theta \mid y_1, \ldots, y_N) = \sum_{n=1}^{N} \ln d(y_n \mid \theta),$$

(9.5)

or the average log-likelihood

$$\frac{1}{N} \sum_{n=1}^{N} \ln d(y_n \mid \theta).$$

(9.6)

The likelihood function can be denoted simply by $L(\theta)$, and its logarithm by $LL(\theta)$. Maximizing the log-likelihood is equivalent to maximize the likelihood since the logarithm operator is monotonically increasing, hence

$$\hat{\theta}_N = \arg \max_{\theta \in \Theta} LL(\theta).$$

$\hat{\theta}_N$ almost surely converges to $\theta_0$ as $N$ grows to infinity, and the distribution function of $\sqrt{N}(\hat{\theta}_N - \theta_0)$ converges to the multinormal distribution function with mean zero and variance-covariance matrix $V$. The reader is referred, e.g., to Newey and McFadden (1994) for more details.

### 9.4 Optimization Algorithms

This section describes algorithms for solving maximum likelihood (or log-likelihood) problems. Such problems are typically expressed as

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f(x) = -L(x)$ or $-LL(x)$, and we use $x$ instead of $\theta$ in order to follow conventional notations in optimization. We then seek optimization algorithms that can reliably converge to a local minimizer from an arbitrary starting point, and do so as quickly as possible. We first
review two classes of widely-used nonlinear optimization methods, i.e., the line search and trust region. Next, we describe different approaches for approximating the Hessian.

9.4.1 Line Search Methods

At each iteration \( k \) of a line search algorithm, we compute a descent direction \( p_k \), i.e. a direction satisfying \( p_k^T \nabla f(x_k) < 0 \), and a positive step length \( \alpha_k \) ensuring a sufficient decrease in the function as measured by the inequality

\[
 f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k,
\]

for some constant \( c_1 \). (9.7) is the first condition of the Wolfe conditions, also called Armijo condition. This condition could be satisfied for all sufficiently small values of \( \alpha \), so it may not be enough by itself to ensure fast convergence, or even convergence to a local solution. Thus, another condition is proposed, called the curvature condition

\[
 \nabla f(x_k + \alpha_k p_k)^T p_k \leq c_2 \nabla f(x_k)^T p_k,
\]

for some constant \( c_2 \) satisfying \( c_1 \leq c_2 < 1 \). (9.8) is sometimes replaced by the strong curvature condition

\[
 |\nabla f(x_k + \alpha_k p_k)^T p_k| \geq c_2 |\nabla f(x_k)^T p_k|,
\]

yielding the strong Wolfe condition. Once a scalar \( \alpha_k \) has been found satisfying the (strong) Wolfe conditions, \( x_{k+1} \) is set to \( x_k + s_k \), where \( s_k = \alpha_k p_k \) is the accepted step at \( k^{th} \) iterate.

In the steepest descent approach, the search direction is simply the opposite of the gradient

\[
p_k = -\nabla f(x_k).
\]

The Newton or quasi-Newton methods compute \( p_k \) by minimizing the predictive quadratic model

\[
m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H_k p,
\]

where \( H_k \) is a positive definite matrix, leading to \( p_k = -H_k^{-1} \nabla f(x_k) \). In the Newton method, we set \( H_k = \nabla^2 f(x_k) \) if the function Hessian is positive definitive, or in other words, if the function is strictly convex at \( x_k \), otherwise we set \( H_k = \nabla^2 f(x_k) + \lambda I \) where \( \lambda \) is large enough to ensure that \( H_k \) is positive definite, but not too large in order to avoid too small steps. In quasi-Newton method, \( H_k \) is an approximation of the Hessian updated at every iteration.
9.4.2 Trust Region Methods

In a trust region algorithm, at each iteration, we solve the subproblem

$$\min_{s \in B_k} m_k(s),$$

(9.10)

where $m_k$ is some model of the objective function and $B_k$, called the trust region, is a region centered at the current iterate, defined as

$$B_k = \{ s \in \mathbb{R}^n \text{ and } ||s||_k \leq \Delta_k \},$$

$\Delta_k$ is the trust region radius and $|| \cdot ||_k$ is a norm, possibly iteration-dependent. An usual choice is the 2-norm, and $m_k$ is typically defined as a quadratic approximation, as in (9.9). The exact minimization of (9.10) is often expensive and unnecessary, so instead it can be solved approximately using for instance the Steihaug-Toint algorithm (Philippe L. Toint, 1981, Steihaug, 1983).

The main idea of trust region methods is then to compare the decrease predicted by the model minimization with the actual decrease of the objective function, by computing the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}.$$

If the agreement $\rho_k$ is sufficiently good, the trial point becomes the new iterate and the trust region is maintained or enlarged. In such a case, the iteration is said to be successful or very successful, depending of the magnitude of $\rho_k$. If this agreement is poor, the trust region is shrunk in order to improve the quality of the model. We refer the reader to Conn et al. (2000b) or Nocedal and Wright (2006, Chapter 4) for more details.

9.5 Hessian Approximations

In order to have a close approximation of the objective function, we should ideally take $H_k$ in (9.9) as the Hessian of the objective function, evaluated at $x_k$. However, as discussed earlier, the computation of the exact Hessian is often too expensive, especially for the discrete choice models, and several approximation methods have been proposed. We now describe some well-known approaches.
9.5.1 Secant Approximations

If the matrix $H_{k+1}$ satisfies the secant equation

$$H_{k+1}s_k = y_k,$$  \hfill (9.11)

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, the quadratic model at iteration $k+1$ enjoys the properties $m_{k+1}(0) = f(x_{k+1})$, $\nabla_x m_{k+1}(0) = \nabla f(x_{k+1})$, and $\nabla_x m_{k+1}(-s_k) = \nabla f(x_k)$, suggesting that the model at iteration $k+1$ is a reasonable approximation of the objective function around $x_{k+1}$. However, we have to impose additional conditions on $H_{k+1}$ as (9.11) does not allow to determine it uniquely.

A first possibility is to require that among all symmetric matrices satisfying the secant equation in some sense, the one closest to the current matrix $H_k$ is selected such that

$$\min_{H=H^T, Hs_k=y_k} \|H - H_k\|_W,$$

where $\| \cdot \|_W$ is the weighted Frobenius norm: $\|A\|_W = \|W^{\frac{1}{2}}AW^{\frac{1}{2}}\|_F$ in which $\| \cdot \|_F$ is defined by $\|C\|_F = \sqrt{\sum_{1\leq i,j\leq n} c_{ij}^2}$. The weight $W$ can be chosen as a matrix satisfying the condition $Wy_k = s_k$. This condition allows an easy solution of the problem above, the unique solution being

$$H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}.$$  \hfill (9.12)

This update is called the BFGS update (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970), and is one of the most popular Hessian approximation used in nonlinear optimization.

Another well-known approximation matrix is the symmetric rank-1 (SR1) update which maintains the symmetry of the matrix but does not guarantee positive definiteness, allowing to take advantage of a negative curvature. The SR1 update also complies with the secant equation in (9.11) with the additional requirement $H_{k+1} = H_k \pm \delta \delta^T$. The only formula that satisfies these conditions is given by (Conn et al., 1991)

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}.$$  \hfill (9.13)

Since this Hessian approximation is not necessarily positive definite, the quadratic model (9.9) can be unbounded below. This is not an issue for trust region methods as the search space is bounded at each iteration, but can lead to failure of line search methods, requiring modifications of the algorithms (Öztoprak and Birbil, 2011).
9.5.2 BHHH Approximation

When maximizing the log-likelihood to estimate model parameters, a specific Hessian approximation can be derived, reflecting the problem structure. If the model is correctly specified and assuming that $\theta_0$ is the true parameters vector, we have the information matrix equality

$$ I(\theta_0) = -E[\nabla^2 d(y|\theta_0)], $$

where $I(\theta_0)$ is the Fisher information matrix, defined as the covariance matrix of the score at $\theta_0$, and the expectation is taken over the population. The score is defined as

$$ g(y|\theta) = \nabla_{\theta} \ln d(y|\theta), $$

leading to the following expression for the information matrix

$$ I(\theta_0) = E[\nabla_{\theta} \ln d(y|\theta_0) \nabla_{\theta} \ln d(y|\theta_0)^T]. $$

For a finite sample, the information matrix can be consistently estimated as

$$ I_N(\theta^*_N) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\theta} \ln d(y_n|\theta^*_N) \nabla_{\theta} \ln d(y_n|\theta^*_N)^T. $$

Berndt et al. (1974) suggest to extrapolate on the information identity, using the opposite of the (sample) information matrix as the Hessian approximation:

$$ H_{BHHH}(\theta) = -I_N(\theta). \quad (9.14) $$

This approximation is also known as the BHHH or statistical approximation. It is positive definite and can be used at each iteration of the trust region or line search algorithms. It only requires the information available at the current iteration, and is cheap to obtain. Moreover, as it relies on the specific properties of the maximum log-likelihood problem, the BHHH approximation is often closer to the true Hessian than the secant approximations, especially during the first iterations. The secant approximations only asymptotically converge, under some conditions, to the true Hessian when the number of iterations is large enough. However, there are two issues that affect the use of the BHHH approximation. First, the information matrix equality is only valid asymptotically with the number of observations, at the true parameters. Second, it requires a correctly specified model, which can be very difficult to obtain. Therefore, the BHHH approximation may not converge to the Hessian of the log-likelihood function, sometimes leading to poor performances, especially when close to the solution.
9.5.3 Corrected BHHH Approximations

For maximum likelihood problems, (9.2) can be rewritten as

\[ \nabla^2 LL(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \frac{\nabla d(y_n|\theta) \nabla d(y_n|\theta)^T}{d^2(y_n|\theta)} + \frac{1}{N} \sum_{n=1}^{N} \frac{\nabla^2 d(y_n|\theta)}{d(y_n|\theta)}. \]  \tag{9.15}

Using (9.14), (9.15) can be rewritten as

\[ \nabla^2 LL(\theta) = H_{BHHH}(\theta) + A(\theta), \]

with

\[ A(\theta) = \frac{1}{N} \sum_{n=1}^{N} \frac{\nabla^2 d(y_n|\theta)}{d(y_n|\theta)}. \]

The computation of \( A(\theta) \) requires the calculation of \( N \) individual Hessian matrices, which may be too expensive. Studies have investigated approximations of \( A(\theta) \) using the secant equation given in (9.11), for instance Dennis, Jr. and Schnabel (1996), Dennis Jr et al. (1981) for least-squares problems, and Bunch (1987) for maximum likelihood estimation. More precisely, assuming that at iteration \( k \) the matrix \( H_k \) is available to approximate the next Hessian \( H_{k+1} \), we can write

\[ H_{k+1} = H_{BHHH}(\theta_{k+1}) + A_{k+1}, \]

where \( A_{k+1} \) is an approximation of \( A(\theta_{k+1}) \). \( A(\theta_{k+1}) \) then can be approximated by two secant equations. First, (9.11) gives \( (H_{BHHH}(\theta_{k+1}) + A_{k+1})s_k = y_k \), and by setting \( \bar{y}_k^1 = y_k - H_{BHHH}(\theta_{k+1})s_k \), this yields the following secant equation

\[ A_{k+1}s_k = \bar{y}_k^1, \]  \tag{9.16}

which can be used to approximate \( A_{k+1} \). Moreover, another second secant equation for \( A_{k+1} \) can be derived by approximating each individual Hessian matrix \( \nabla^2 d(y_n|\theta) \). More precisely, we have

\[ \nabla^2 d(y_n|\theta_k)s_k \approx \nabla d(y_n|\theta_{k+1}) - \nabla d(y_n|\theta_k). \]

Substitution into (9.15) gives

\[ A(\theta_k)s_k \approx \frac{1}{N} \sum_{n=1}^{N} \frac{\nabla d(y_n|\theta_{k+1}) - \nabla d(y_n|\theta_k)}{d(y_n|\theta_k)}. \]

So if we define \( \bar{y}_k^2 = \frac{1}{N} \sum_{n=1}^{N} \frac{\nabla d(y_n|\theta_{k+1}) - \nabla d(y_n|\theta_k)}{d(y_n|\theta_k)} \), then the second secant equation can be written as

\[ A_{k+1}s_k = \bar{y}_k^2. \]  \tag{9.17}
$A_{k+1}$ can be approximated by any secant updates, e.g. the BFGS or SR1, and it is possible to establish the convergence under mild conditions (Dennis Jr and Walker, 1981).

### 9.6 Switching Strategies

The objective of this work is to obtain computationally efficient optimization algorithms for solving maximum likelihood problems. We propose approaches based on the line search and the trust region methods. As previously discussed, several Hessian approximation methods are available, with performances that may vary in different phases of the optimization process. Some authors have considered switching among Hessian approximations during the optimization. For example, Phua and Setiono (1992) proposed a switching algorithm based on the condition number of the secant approximation matrices, while Al-Baali et al. (2004) examine the possibility to switch between the BFGS and SR1 approximations. Dennis Jr et al. (1981) propose switching approaches designed specifically for least-squares problems, and rely on the trust region algorithm and the Davidon-Fletcher-Powell (DFP) method to compute the correction term. Bunch (1987, 1988) applied this approach for the maximum likelihood estimation, switching between two quadratic models, one uses the BHHH and an alternative model using a corrected BHHH approximation. Al-Baali (1984) suggested to use a correction based on the BFGS update for nonlinear least-squares, the convergence of this approach was formally proved by Dennis Jr et al. (1989), and extended by Zhou and Chen (2010).

In this section, we develop a simple and general framework that allows the optimization algorithms (line search or trust region) to switch between more than two quadratic models based on different available Hessian approximations at each iteration. At first we present, in the following, the general framework so that our algorithms can be described explicitly.

We denote by $\mathcal{H}_k$ the set of available Hessian approximations to select from at the $k^{th}$ iteration of an optimization algorithm

$$\mathcal{H}_k = \{H^i_k, i = 1, \ldots\},$$

where $H^i_k$ refers to a specific Hessian approximation. For example, the BHHH can be denoted by $H^1_k$ and the BFGS can be denoted by $H^2_k$, etc. Each iteration of an optimization algorithm with a switching strategy executes one more step in which one Hessian approximation is chosen from $\mathcal{H}_k$ in order to compute the search direction (for line search) or to define the subproblem (for trust region). The next two sections describe our switching strategies.
9.6.1 Multi-subproblems Method

At iteration $k$, given a set of Hessian approximations $\mathcal{H}_k$, there is a set of the corresponding subproblems as

$$\min_{s \in B_k} m_i^k(s) = \min_{s \in B_k} \left\{ f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T H_k^i s \right\}, \quad H_k^i \in \mathcal{H}_k. \tag{9.18}$$

A classical trust region algorithm solves one subproblem at each iteration, and a step is determined to be successful based on the decrease of the objective function and the decrease of the corresponding quadratic model. Given several subproblems available at each iteration, a number of steps can be computed. Since we aim at going as quickly as possible to the optimum (in terms of number of iterations), the multi-subproblems method is designed to select the step that reduces the most the objective function. More precisely, we solve approximately all the available subproblems in order to obtain the set of steps $\{s_i^k\}$ and to choose a step $s_i^k$ which satisfies

$$i^* \in \arg\min_i f(x_k + s_i^k). \tag{9.19}$$

A trust region method with the multi-subproblems switching strategy is described in Algorithm 1. We note that Algorithm 1 requires solving all the subproblems, therefore calculating more than one objective function value at each iteration. More precisely, the number of objective function values evaluated at each iteration is equal to the number of quadratic models considered. This approach therefore may be more expensive in terms of computational time, but we expect that it may require a lower of number of iterations, compared to other approaches.

9.6.2 Predictive Method

The approach proposed by Dennis Jr et al. (1981) allows to switch between two Hessian approximations at each iteration. The idea is to predict a model for the next iteration by comparing the differences between the objective values and the values given by quadratic models. This switching strategy, as mentioned above, is designed specifically for the trust region with only two Hessian approximations. In this section we adapt this approach and develop a more flexible predictive method that allows to switch between several Hessian approximations. Moreover, it can be easily integrated into both trust region and line search algorithms.

Consider a set of models as in (9.18) and denote by $\delta_k^i(s)$ the absolute difference between the quadratic model $m_k^i(s)$ and $f(x_k + s)$. We call $\delta_k^i(s)$ the approximation error of the quadratic model $m_k^i(s)$. The predictive method uses this quantity to evaluate the accurateness of the quadratic model and to select a Hessian approximation for the next iteration. More precisely, at the end of an iteration $k$ with step $s_k$ of some optimization algorithm (either trust region of line search), and given that the objective function $f(x_k + s_k)$ is already computed, the
Algorithm 1 Trust region method with the multi-subproblems switching model

**Step 0. Initialization:** Given an initial point $x_0$, an initial trust region with radius $\Delta_0$ and constants $\eta_1, \eta_2, \gamma$ which satisfy

$$1 > \eta_1 > \eta_2 > 0 \text{ and } 1 > \gamma_1 > \gamma_2 > 0,$$

choose an initial matrix $H_0$ and set $k = 0$.

**Step 1.** If stopping criteria are met, stop. Otherwise, go to Step 2.

**Step 2.** Define a set of Hessian approximations $H_k$.

**Step 3. Step calculation:** Calculate the set of steps $\{s_i^k, i = 1, 2, \ldots\}$ by solving approximately all the subproblems.

$$\min_{s \in B_k} \{m^i(s), H^i_k \in H_k\}.$$

Determine the best step $s^*_k$ by solving (9.19).

Compute the ratio $\rho_k$

$$\rho_k = \frac{f(x_k) - f(x_k + s^*_k)}{f(x_k) - m_k(s^*_k)}.$$

If $\rho_k > \eta_2$ set $x_{k+1} = x_k + s_k$, otherwise set $x_{k+1} = x_k$.

**Step 4. Trust region radius update:** Update the trust region radius as follows:

$$\Delta_{k+1} = \begin{cases} \max\{2\|s_k\|, \Delta_k\} & \text{If } \rho_k \geq \eta_1 \\ \gamma_1 \Delta_k & \text{If } \eta_1 > \rho_k \geq \eta_2 \\ \gamma_2 \Delta_k & \text{If } \rho_k \leq \eta_2 \end{cases}$$

Set $k \leftarrow k + 1$ and go to Step 1.

Approximation errors associated with different Hessian approximations can be computed as

$$\delta^i_k(s_k) = |f(x_k + s_k) - m^i_k(s_k)| = \left| f(x_k + s_k) - f(x_k) - s^T_k \nabla f(x_k) - \frac{1}{2} s^T_k H^i_k s_k \right|.$$

Consequently, the next Hessian approximation $H^*_{k+1}$ is predicted by minimizing this error

$$i^* = \arg\min_i \{\delta^i_k(s_k)\}.$$  \hspace{1cm} (9.20)

The predictive switching strategy has the advantage of not requiring any new evaluation of the objective function, which is often costly to evaluate, particularly with large real data sets. Avoiding the evaluation of this function improves the computational efficiency of the optimization methods. A trust region algorithm with the predictive switching strategy is described in Algorithm 2. A line search algorithm with the predictive switching strategy is described in Algorithm 3.
Algorithm 2 Trust region method with the predictive switching model

Steps 0–2. Identical to Algorithm 1.

Step 3. Step calculation: Evaluate the step $s_k$ by solving approximately the subproblem

$$
\min_{x_k + s \in B_k} m_k(s).
$$

Evaluate $\rho_k$

$$
\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)}.
$$

If $\rho_k > \eta_2$ set $x_{k+1} = x_k + s_k$, otherwise set $x_{k+1} = x_k$.

Step 3. Hessian approximation prediction: The next Hessian approximation $H^{i*}_{k+1}$ is predicted by solving (9.20). Set $H_{k+1} = H^{i*}_{k+1}$.

Step 4. Trust region radius update: Identical to Step 4 of Algorithm 1. Set $k \leftarrow k+1$ and go to Step 1.

Algorithm 3 Line search method with the predictive switching model

Step 0. Initialization: Choose an initial matrix $H_0$ and set $k = 0$.

Step 1. If stopping criteria are met, stop. Otherwise, go to Step 2.

Step 2. Search direction calculation: Compute search direction $p_k$ which satisfies the equation:

$$
H_k p_k = -\nabla f(x_k).
$$

Step 3. Step calculation: Compute step length $\alpha_k$ which satisfies the Wolfe conditions and set $x_{k+1} = x_k + \alpha_k p_k$.

Step 4. Hessian approximation prediction: Predict the next Hessian approximation $H^{i*}_{k+1}$ using (9.20). Set $H_{k+1} = H^{i*}_{k+1}$.

Set $k \leftarrow k+1$ and go to Step 1.

9.7 Discrete Choice Theory

The proposed switching techniques have been applied on the estimation of various discrete choice models, so before describing our numerical experiments, we provide a short introduction to discrete choice models and applications related to the results presented in this paper.

9.7.1 Discrete Choice Models

Discrete choice theory examines how people make decisions among a finite number of possible choices. More specifically, we consider a set of $N$ individuals, each one has to choose one alternative within a finite set. The Random Utility Maximization (RUM) theory (McFadden,
(1973) assumes that each individual $n$ associates to each alternative $i$ within a choice set $C_n$ an utility $U_{ni}$. This utility consists of two parts: a deterministic part $V_{ni}$ known by the modeler and an uncertain part $\epsilon_{ni}$ which is known to individuals but unknown to modelers. The utility is

$$U_{ni} = V_{ni} + \epsilon_{ni}.$$  

The deterministic term $V_{ni}$ can include attributes of the alternative as well as socio-economic characteristics of the individual. In general a linear-in-parameters is used, i.e., $V_{ni} = \beta^T x_{ni}$, where $\beta$ is a vector of parameters to be estimated and $x_{ni}$ is the vector of attributes of alternative $i$ observed by individual $n$. The decision maker aims to maximize the utility so the probability that an alternative $i$ is chosen by individual $n$ is $P(i|n, C_n) = P(U_{ni} \geq U_{nj}, \forall j \in C_n)$. Different assumptions for the random terms $\epsilon_{nj}, j \in C_n$ can lead to different types of discrete choice models. A popular model is the multinomial logit (MNL) which assumes that the random terms are independent and identically distributed (i.i.d.) Extreme Value type I with mean $\mu$ and scale factor $\lambda$ (often normalized to one), characterized by the distribution function $F(x) = e^{-e^{-\lambda(x-\mu)}}$, the choice probability is then

$$P_L(i|n, C_n) = \frac{e^{\lambda V_{ni}}}{\sum_{j \in C_n} e^{\lambda V_{nj}}}.$$  \hspace{1cm} (9.21)

Such model can be estimated by maximizing the log-likelihood function over the parameters vector $\beta$

$$\max_{\beta} LL(\beta) = \frac{1}{N} \sum_{n=1}^{N} \ln P_L(i|n, C_n).$$

The MNL model provides a simple closed form for the choice probabilities. It however has an important limitation which is the independence of irrelevant alternatives (IIA) property (for instance Train, 2003). Other models have been proposed in order to relax this limitation. Examples are the nested logit model (Ben-Akiva, 1973), the mixed logit model (McFadden, 1978) and the cross-nested logit models (Vovsha and Bekhor, 1998). In the following we briefly describe the mixed logit and logit route choice models, which are used in our numerical tests.

### 9.7.2 Mixed Logit Models

Mixed logit models have been known for many years but have become popular with McFadden and Train (2000). They obviate the three limitations of the MNL model by allowing for random taste variation, unrestricted substitution patterns, and correlation in unobserved factors over time (Train, 2003). Mixed logit models can be derived under a variety of different behavioural specifications, where each derivation provides a particular interpretation. The first application of mixed logit models was apparently the demand for electricity-using goods (Electric Power Research Institute, 1977).
Using the random-terms formulation, we here assume that the vector of model parameters $\beta$ is itself derived from a random vector $\omega$ and a parameter vector $\theta$, which we express as $\beta = \beta(\omega, \theta)$. $\omega$ typically specifies the random nature of the model and the vector parameters $\theta$ quantifies the population characteristic for the model. The associated unconditional probability is obtained by integrating over $\omega$ as

$$
P_{ML}(i|n, C_n, \theta) = E_P[P_L(i|n, C_n, \omega, \theta)] = \int P_L(i|n, C_n, \omega, \theta)f(\omega)d\omega,
$$

(9.22)

where $P$ is the probability measure associated to $\omega$, $E$ is the expectation operator and $f$ is the density function. When $T_n$ observations are available per individual, the correlation is often captured by assuming that the parameters $\beta$ do not vary for the same individual, while being randomly distributed throughout the population (Revelt and Train, 1998). (9.22) then becomes

$$
P_{ML}(i|n, C_n, \theta) = E_P \left[ \prod_{t=1}^{T_n} P_L(i_t|n, C_n, \omega, \theta) \right],
$$

(9.23)

where $i_t$ is the $t^{th}$ observed choice.

It is usual to replace the expectation by an approximation, typically obtained by sampling over $\omega$. (9.23) becomes

$$
P_{ML}(i|n, C_n, \theta) \approx SP_{ML}R(I|n, C_n, \theta) = \frac{1}{R} \sum_{r=1}^{R_n} \prod_{t=1}^{T_n} P_L(i_t|n, C_n, \omega_{r_t}, \theta),
$$

where $R_n$ is the number of random draws associated with individual $n$. The sample can be generated by standard Monte Carlo or quasi-Monte Carlo techniques, though there is no clear advantage of one or the other approach in this context (Munger et al., 2012).

### 9.7.3 Path-based Route Choice Models

Discrete choice models are also used for analyzing and predicting route choices in various transport applications. The route choice problem using discrete choice models is characterized by a very large number of path alternatives and it is not possible to enumerate all paths connecting a given origin-destination pair in a real network. In order to consistently estimate MNL route choice models, Frejinger et al. (2009) suggest that paths can be sampled and the choice probabilities can be corrected using sampling probabilities. We briefly present this sampling approach in this section, as it is related to our numerical tests.
When the path choice sets are sampled, for each individual \( n \) and a sampled choice set \( D_n \), the probability that a path \( \sigma \) is chosen is

\[
P(\sigma|n, D_n) = \frac{e^{V_{n\sigma} + \ln \pi(D_n|\sigma)}}{\sum_{j \in D_n} e^{V_{nj} + \ln \pi(D_n|j)}},
\]

where \( V_{nj} \) is the deterministic utility of path \( j \) observed by the individual \( n \) and \( \ln \pi(D_n|j) \) is the correction for sampling bias. \( \pi(D_n|j) \) is the probability of sampling choice set \( D_n \) given that \( j \) is the chosen alternative (Frejinger et al., 2009). Mai et al. (2015d) show that when the models are correctly specified, the information matrix equality holds if and only if the sampling corrections \( \pi(D_n|j), j \in D_n \), are added to the choice probabilities.

In order to deal with the overlapping of paths in the network, Frejinger et al. (2009) propose a heuristic sampling correction of the path size attribute (Ben-Akiva and Bierlaire, 1999a), called the expanded path size (EPS). With the EPS attribute, the path choice probability becomes

\[
P(\sigma|D_n) = \frac{e^{V_{n\sigma} + \beta \text{EPS}_n(\sigma) + \ln \pi(D_n|\sigma)}}{\sum_{j \in D_n} e^{V_{nj} + \beta \text{EPS}_n(\sigma) + \ln \pi(D_n|j)}},
\]

where \( \text{EPS}_n(\sigma) \) is the EPS attribute of path \( \sigma \) observed by individual \( n \). The EPS attribute can be computed based on the lengths of links lying on the corresponding path and an expanded factors (Frejinger et al., 2009).

We note that the MNL model is used in many route choice applications in spite of the fact that error terms are believed to be correlated due to the physical overlap among paths, and the EPS attribute allows to partly address this issue. If the random terms are still correlated, the model is misspecified, leading to potential issues when relying on the BHHH technique. Modeling issues can also be presented in the deterministic part of the utilities. The reader can consult Mai et al. (2015d) for a discussion about the invalidity of the information matrix equality for MNL route choice models.

### 9.7.4 Link-based Recursive Logit Route Choice Models

The link-based recursive logit (RL) is a route choice model proposed by Fosgerau et al. (2013a), that can be consistently estimated without sampling of choice sets and quickly used for prediction. In the RL model, a path choice is modeled as a sequence of link choices based on the dynamic discrete choice framework proposed by Rust (1987). Fosgerau et al. (2013a) show that the RL is equivalent to a MNL route choice model based on the universal choice sets, i.e., the choice sets of all feasible paths in the network connecting given origin-destination pairs. The estimation of the RL is considerably more expensive than other path-based models with sampled choice sets (ignoring the computational time associated with sampling), as it requires solving
several dynamic programming problems. Beside the path-based models mentioned above, we also use the RL model for our numerical tests. Moreover, in order to challenge our optimization algorithms, we consider the more complicated RL-based models proposed in Mai et al. (2015b), based on the random regret minimization principle (RRM) (Chorus, 2010).

In the RL model with the RRM framework, given two connected links \(a, k\), we associate the following instantaneous random regret for individual \(n\)

\[
rr_n(a|k) = r_n(a|k) + \mu \epsilon_n(a),
\]

where \(r_n(a|k)\) is the deterministic part of the link regret associated with link \(a\) given \(k\), \(\epsilon_n(a)\) are i.i.d. extreme value type I distributed error terms and \(\mu\) is a strictly positive scale parameter.

Mai et al. (2015b) propose different RRM-based RL models based on the generalized RRM model (Chorus, 2014), but in this paper we consider the original version proposed in Chorus (2010), so that the resulting estimation problem becomes unconstrained. More precisely, the deterministic part of the link regret function associated with link \(a\) given \(k\) is defined as

\[
r_n(a|k) = \sum_{a' \in A(k)} \sum_t \ln \left( 1 + e^{\beta_t (x(a'|k)_t - x(a|k)_t)} \right),
\]

(9.24)

where \(A(k)\) is the set of outgoing links from the sink node of \(k\), \(x(a|k)\) is a vector of attributes associated with link \(a\) given \(k\), \(t\) is an attribute, and \(\beta\) is the vector of parameters to be estimated. Indeed, the link regret functions are nonlinear-in-parameters, hence the RL-RRM model becomes more expensive to estimate than the linear-in-parameter random utility maximization (RUM) based RL model proposed in Fosgerau et al. (2013a).

### 9.8 Numerical Assessment

#### 9.8.1 Data Sets

In order to evaluate the performance of the various considered optimization algorithms, we estimate models on three real data sets, i.e., two sets used with mixed models (SP2 and IRIS) and one data set feeding two path-based route choice models (PL and PSL) and one link-based model (RL-RRM). Table 9.1 provides the numbers of observations, along with the numbers of individuals, the numbers of alternatives, and the numbers of model parameters. We now describe these data sets briefly.
The discrete choice data tests have been conducted on two real data sets: Cybercar (Cirillo and Xu, 2010) and IRIS (Bastin et al., 2010). Cybercar is a data set that has been collected in April 2008 at the Baltimore/Washington International Airport and concerns the operation of an automated vehicle technology called Cybercars. Our tests utilize only part of this data set which we refer to as SP2. IRIS refers to a regional transport model in Belgium where data have been collected on the propensity to switch from car to public transportation. Seven of the explanatory variables in the IRIS model are randomly distributed, with two of them assumed to be normal or log-normal (congested and free flow time coefficients) and the remaining five are assumed to be normal. When the congested and free flow time coefficients have normal distribution we identify this model as the IN model, when they have a log-normal distribution the model is identified as the ILN model.

The route choice data was collected on the Borlänge network in Sweden which is composed of 3077 nodes and 7459 links. The path sample consists of 1832 trips corresponding to simple paths with a minimum of five links. There are 466 destinations, 1420 different origin-destination (OD) pairs and more than 37,000 link choices in the sample. The route choice data were collected by GPS monitoring, therefore the socio-economic information about the drivers is not available. We note that the same data have been used in other route choice modeling articles (Fosgerau et al., 2013a, Frejinger, 2007, Mai et al., 2015c). We use the link-based model (RL-RRM) and two path-based models with and without the EPS attribute (denoted by PL and PSL, respectively). For the path-based models, for each observation we sample a choice set of 50 draws. See Frejinger et al. (2009) and Mai et al. (2015b,d) for details on the model specifications and estimation results.

9.8.2 Performance Comparisons

We compare the performance of switching approaches with trust region and line search algorithms using a single Hessian approximation. In our experiments, the numbers of the model
parameters are small, hence the cost of model minimization in trust region methods or search direction computation in line search algorithms is typically negligible, compared to the evaluation cost of the log-likelihood. Indeed, the problem dimension is usually small while the number of observations is large, as shown in Table 9.1. Therefore, the number of objective function evaluations captures most of the computational cost, and will be used to compare performance among algorithms. As the purpose of the comparisons is to evaluate the impact of different Hessian approximations on the performance of the optimization algorithms, the estimation results as well as the analysis on the effects of the Monte-Carlo methods will not be reported. Note that 1000 Monte Carlo random draws per individual are used for the mixed logit models. We use the Monte Carlo method to sample choice sets for the estimation of the PL and PSL models (see for instance Mai et al., 2015d). All the reported numerical results for the mixed, PL and PSL models are based on 10 independent simulations. The numerical evaluations for the mixed logit models have been carried out using the package AMLET (Bastin et al., 2006a). The optimization algorithms to estimate the route choice models were implemented in MATLAB. We note that the Decomposition method (Mai et al., 2015b) is used to estimate the RL-RRM model, as suggested by Mai et al. (2015b).

When a single Hessian approximation is used, trust region algorithms are implemented either with the BHHH, BFGS or SR1 Hessian approximation, and line search algorithms are implemented either with the BHHH or BFGS. Line search algorithms have not been implemented with SR1 as it does not guarantee descent search directions. For the switching approaches, we have implemented trust region algorithms with the multi-subproblems and the predictive methods, and a line search algorithm with the predictive method. In order to facilitate our discussions, we denote by $BHHH_{corr1}^{-}\text{BFGS}$ and $BHHH_{corr2}^{-}\text{BFGS}$ the corrected BHHH approximations using respectively (9.16) and (9.17), and where $A_{k+1}$ is updated by the BFGS method. We denote by $BHHH_{corr1}^{-}\text{SR1}$ and $BHHH_{corr2}^{-}\text{SR1}$ the SR1 approximations based respectively on (9.16) and (9.17).

We use the strong Wolfe conditions to compute the search directions for line search algorithms while the sub-problems are solved by the Steihaug-Toint algorithm. For the line search algorithms, the chosen parameters for the strong Wolfe conditions are $c_1 = 10^{-4}$ and $c_2 = 0.9$. It is also a typical choice in practice. For the trust region algorithms, different parameters are chosen between the mixed logit and route choice models in order to obtain better performance (i.e. lower number of iterations) in each context. Parameters $\eta_1 = 0.9$, $\eta_2 = 0.01$ are chosen for the mixed logit models and $\eta_1 = 0.75$, $\eta_2 = 0.05$ for route choice models. We also assign $\gamma_1 = 0.7$, $\gamma_2 = 0.5$ for all the models.

Each iteration of our switching methods allows to switch between several Hessian approximations, hence, the specification of a set of Hessian approximations for each switching method
is important. For the multi-subproblems method, as the number of function evaluations depends on the number of matrices in the set, we only select two approximations: BHHH and BHHH\textsuperscript{corr1}-BFGS. For the trust region combined with the predictive method, we select three Hessian approximations: the BHHH, BHHH\textsuperscript{corr1}-BFGS and BHHH\textsuperscript{corr1}-SR1. We note that the BHHH\textsuperscript{corr1}-SR1 is chosen in order to take advantage of a negative curvature, and this choice always outperforms the predictive method with only BHHH and BHHH\textsuperscript{corr1}-BFGS in our experiments. Moreover, the BHHH and BHHH\textsuperscript{corr1}-BFGS approximations have been selected for the line search algorithm with the predictive methods, as the SR1 does not always produce a descent direction. We note that the performance of the BHHH\textsuperscript{corr1} and BHHH\textsuperscript{corr2} are very similar, therefore they are not included in a same set of Hessian approximations.

Considering the two switching methods, the two optimization framework (trust region and line search), and several Hessian approximations, we estimate the mixed logit and logit-based route choice models using the following optimization algorithms:

1. **TR-BHHH**: Trust region algorithm with BHHH
2. **TR-BFGS**: Trust region algorithm with BFGS
3. **TR-SR1**: Trust region algorithm with SR1
4. **TR-PRED**: Trust region algorithm with the predictive method
5. **TR-MULTI**: Trust region algorithm with the multi-subproblems method
6. **LS-BHHH**: Line search algorithm with the BHHH
7. **LS-BFGS**: Line search algorithm with the BFGS
8. **LS-PRED**: Line search algorithm with the predictive method

We stop any algorithm if one of the conditions described in Table 9.2 is satisfied, declaring a success or a failure depending on the case encountered. Parameters MAX-ITER = 300 and $\epsilon = 10^{-5}$ were chosen for all the algorithms.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Stopping test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla f(x_k) \leq \epsilon$</td>
<td>GRADIENT</td>
<td>Successful</td>
</tr>
<tr>
<td>$\nabla f(x_k) \geq$ max, $(\frac{</td>
<td></td>
<td>\nabla f(x_k)</td>
</tr>
<tr>
<td>$k \geq$ MAX-ITER</td>
<td>TOO MANY ITERATIONS</td>
<td>Fail</td>
</tr>
<tr>
<td>$\theta &lt; x_{k+1} - x_k \leq \epsilon$</td>
<td>STEP SIZE</td>
<td>Fail</td>
</tr>
<tr>
<td>$\Delta_k \leq \epsilon$</td>
<td>TRUST REGION RADIUS</td>
<td>Fail</td>
</tr>
</tbody>
</table>

**Table 9.2**: Summary of stopping conditions

Tables 9.3 and 9.4 report average numbers of function evaluations for all the algorithms. For the sake of comparison we report the average number of iterations in parentheses and the number of
failures in brackets. The quantities are reported based only on successful runs and the number in bold in each column is the best result. Among trust region algorithms with a single Hessian approximation, the results show that for the SP2 and IN models, the BHHH approximation performs better than the BFGS and SR1), which explains why BHHH is often the favorite approach for MLE. For these models, the algorithms with the BHHH method always reach the optimal solution rapidly. Alternatively, the BFGS and SR1 perform significantly better than the BHHH for the route choice models when used in the trust region algorithm. This can partly be explained by the violation of the information matrix equality, which has been shown in Mai et al. (2015d). For the most complex model, i.e. ILN, the TR-BHHH algorithm has failed to converge for 9 of the 10 runs. In this case, the algorithm rapidly converges to the neighborhood of the solution, but then progresses very slowly close to the optimum, finally it fails to satisfy one of the successful stopping conditions. The norms of the gradient and relative gradient are always greater than $10^{-4}$ since the threshold for our algorithms is $\epsilon = 10^{-5}$. We however note that for the successful case the TR-BHHH performs the best compared to the other algorithms. These results translate a well-known behavior of the BHHH approximation as it may not converge to the true Hessian due to misspecification issues. On the contrary, there are no failures for the line search algorithms, even for the ILN model. Line search algorithms present the same trends than trust region methods between BFGS and BHHH, but are sometimes faster, sometimes slower.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>SP2</th>
<th>IN</th>
<th>ILN</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR-BHHH</td>
<td>27.0</td>
<td>23.9</td>
<td>37.0*</td>
</tr>
<tr>
<td>TR-BFGS</td>
<td>52.9</td>
<td>155.1</td>
<td>147.6</td>
</tr>
<tr>
<td>TR-SR1</td>
<td>42.1</td>
<td>241.5</td>
<td>238.4*</td>
</tr>
<tr>
<td>TR-PRED</td>
<td>14.2</td>
<td>21.8</td>
<td>54.7</td>
</tr>
<tr>
<td>TR-MULTI</td>
<td>46.4</td>
<td>40.4</td>
<td>77.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>PS</th>
<th>PSL</th>
<th>RL-RRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR-BHHH</td>
<td>40.5</td>
<td>58.2</td>
<td>123</td>
</tr>
<tr>
<td>TR-BFGS</td>
<td>19.6</td>
<td>22.5</td>
<td>34</td>
</tr>
<tr>
<td>TR-SR1</td>
<td>24.5</td>
<td>25.4</td>
<td>29</td>
</tr>
<tr>
<td>TR-PRED</td>
<td>20.6</td>
<td>19.6</td>
<td>65/38</td>
</tr>
<tr>
<td>TR-MULTI</td>
<td>33.2</td>
<td>31.4</td>
<td>102/51</td>
</tr>
</tbody>
</table>

Table 9.3: Performance comparison with mixed logit models

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>PS</th>
<th>PSL</th>
<th>RL-RRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-BHHH</td>
<td>22.6</td>
<td>22.2</td>
<td>61</td>
</tr>
<tr>
<td>LS-BFGS</td>
<td>22.6</td>
<td>22.2</td>
<td>61</td>
</tr>
<tr>
<td>LS-PRED</td>
<td>22.6</td>
<td>22.2</td>
<td>61</td>
</tr>
</tbody>
</table>

Table 9.4: Performance comparison with MNL route choice models
Among the switching algorithms, the predictive method with line search and trust region algorithms is slightly better than other classical algorithms, except for the RL-RRM model. In this case, the BTR-PRED and LS-PRED are significantly slower than other classical algorithms with single Hessian approximations, i.e. the BTR-BGFS, BTR-SR1 or LS-BFGS. This can be explained by looking at the performance of the BHHH method. Indeed, the BHHH method performs the worst, compared to the other Hessian approximations, and in fact all the Hessian approximations used in the predictive method are based on the BHHH (i.e. the BHHH or corrected BHHH). In order to verify if it is the case, we include the BFGS and SR1 into the set of Hessian approximations for the BTR-PRED, and include the BFGS for the LS-PRED. We then report the performance of the predictive method via the numbers in italics (Table 9.4). Indeed, the performance of the BTR-PRED and LS-PRED is improved, and becomes competitive with the BFGS and SR1 methods. This result also indicates the flexibility of the predictive method, in the sense that it allows to extend the set of Hessian approximations to improve the optimization algorithms. This flexibility is therefore an advantage of our predictive approach, compared to other classical switching strategies used in Dennis Jr et al. (1981) and Bunch (1987).

The results show that the predictive algorithms are always competitive, both for the trust region and the line search version, while the TR-MULTI is the slowest. This is expected as the TR-MULTI algorithm requires two evaluations of the objective function at each iteration, leading to double the number of function evaluations, while the other trust region strategies only compute the objective function once per iteration. In other words, the TR-MULTI is usually more effective in terms of iterations, while the total cost, involving the cost per iteration, is higher.

In all the previous experiments, standard starting points ($x_0 = 0$ for the mixed logit models and $x_0 = (-3, \ldots, -3)$ for the PL and PSL models) are chosen as the initial point of the iterative process. Exceptionally, the starting point for the RL-RRM model is $x_0 = (-1.5, -4.0, -1.0, -3.5)$. This point is chosen manually so that the dynamic programing problem given by the RL-RRM has a solution. To evaluate the performance of the algorithms in difficult cases, we perform additional experiments using the ILN, the most complex model, with a starting point chosen far away from the optimum. Table 9.5 reports the success rate of the simulations for ILN when the initial vector of parameters is unusual $x_0 = (20.0, -25.0, -20.0, 13.0, 21.0, 30.0, -14.0, -21.0, -13.0, -1.0, 31.0, -8.0, -22.0, 0.0, 4.0, -32.0, 11.0, -11.0, 32.0, -1.5, 12.0, 15.2, -11.5, -0.6, 32.7)$. Note that the optimal parameter of this model is $\hat{x} \approx (-1.1, -5.5, 4.9, -7.3, 6.5, -0.64, -2.8, 1.0, -2.97, -1.10, 0.27, -0.52, 0.216, 0.24, 3.21, -1.14, -1.84, -3.28, -2.83, -2.45, 2.51, -2.71, 1.86, 1.37, 1.91)$, where the optimal log-likelihood value is approximately $-3.15$, much higher than the initial log-likelihood value of $-275.28$.

In Table 9.5, the TR-MULTI algorithm has a success rate of 50%, a clear dominance over the other algorithms. Simulations fail in Table 9.5 mostly on the “STEP SIZE” condition (see Table
where algorithms stop at points which are very far from the optimal solution. We also observed some failures due to the “TOO MANY ITERATIONS” condition (each estimation was limited to 500 iterations). Interestingly, two failure cases due to the “TOO MANY ITERATIONS” condition had a final log-likelihoods very close to the optimum. These failures belong to the TR-BHHH algorithm.

All the line search algorithms fail to compute a step size that satisfies the strong Wolfe conditions at the beginning of the optimization process (after few iterations only). This observation suggests that the trust region algorithms are more robust than the line search algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Successful cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust region</td>
<td></td>
</tr>
<tr>
<td>TR-MULTI</td>
<td>5/10</td>
</tr>
<tr>
<td>TR-PRED</td>
<td>3/10</td>
</tr>
<tr>
<td>TR-BHHH</td>
<td>0/10</td>
</tr>
<tr>
<td>TR-BFGS</td>
<td>1/10</td>
</tr>
<tr>
<td>TR-SR1</td>
<td>0/10</td>
</tr>
<tr>
<td>Line search</td>
<td></td>
</tr>
<tr>
<td>LS-BHHH</td>
<td>0/10</td>
</tr>
<tr>
<td>LS-BFGS</td>
<td>0/10</td>
</tr>
<tr>
<td>LS-PRED</td>
<td>0/10</td>
</tr>
</tbody>
</table>

Table 9.5: Rate of successful simulations for a difficult case (ILN)

Finally, in Table 9.6 we report the number (or average number) of switching for our three switching models TR-PRED, TR-MULTI and LS-PRED. In this table we observe a small average number of switches for the LS-PRED, which means that LS-PRED often uses BHHH during the all the optimization process or switched one or two times to a corrected BHHH and then uses a fixed approximation scheme until convergence. The contrast with the number of switches in the trust region methods can be partly explained as the possibility for switching is considered once per iteration, but the trust region methods typically requires more, but cheaper, iterations than line search techniques. Moreover, the additional efforts made at each iteration of the line search algorithm to satisfy the Wolfe conditions provide a step that is often more efficient for the model under consideration, limiting the potential for other models to provide a higher function decrease at the new iterate. This suggests that the trust region approach is more suited for switching strategies. Furthermore, for the RL-RRM model, the numbers in italics refer to the number of switches when the BFGS and SR1 are included into the set of Hessian approximations. Indeed, with the larger set of matrices, the performance of the corresponding algorithms is improved, and the number of switches are also larger.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>SP2</th>
<th>IN</th>
<th>ILN</th>
<th>PS</th>
<th>PSL</th>
<th>RL-RRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust region</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR-PRED</td>
<td>5.3</td>
<td>4.7</td>
<td>18.4</td>
<td>6.4</td>
<td>5.7</td>
<td>3/15</td>
</tr>
<tr>
<td>TR-MULTI</td>
<td>6.5</td>
<td>7.8</td>
<td>11.5</td>
<td>8.4</td>
<td>7.7</td>
<td>5</td>
</tr>
<tr>
<td>Line search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS-PRED</td>
<td>1.4</td>
<td>1.0</td>
<td>0.9</td>
<td>1.0</td>
<td>1.0</td>
<td>1/1</td>
</tr>
</tbody>
</table>

Table 9.6: Average number of switches per estimation in the switching methods
9.9 Conclusion

In this paper, we have reviewed standard trust region and line search algorithms for maximum likelihood estimation, with emphasis on the use of Hessian approximation methods to determine the step at each iteration. We have explored the possibility of switching between various Hessian approximations throughout the optimization process. In particular, we proposed the predictive approach, aiming to determine the most suited model between several quadratic approximations at each iteration. This approach does not require any new computation of the objective function. We have also proposed the multi-subproblems method which is based on the fact that, at each iteration of a trust region algorithm, we can solve more than one sub-problem to better determine a step. This approach however requires additional evaluations of the objective function which depends on the size of the set of Hessian approximations considered at each iteration.

We have applied our algorithms to mixed logit and logit based route choice models based on real data sets. The predictive method outperforms the classical optimization algorithms. The multi-subproblems requires large numbers of function evaluations but it has the highest successful rates when solving a complex mixed logit model with an unusual starting point.

In future research we plan to extend further the switching models to use information from several iterations to improve the accurateness of the switching strategies, and to combine this approach with adaptive sampling strategies for mixed logit models (Bastin et al., 2006a). We also emphasize that the proposed algorithms can also be applied to least square problems, extending the method proposed by (Dennis Jr et al., 1981). Moreover, we plan to extend the switching algorithms to other classes of nonlinear optimization problems.
Chapter 10

Conclusions and Future Research
Perspectives

This thesis is based on a collection of seven articles. Two of them are published in international journals, four have been submitted for possible publication, and the last one will be submitted soon. In this chapter we summarize the main results and directions for future research.

10.1 Conclusions

In this thesis we proposed a number of models and methods related to modeling correlation in the route choice problem. The work builds on the ideas from Fosgerau et al. (2013a) where a new way to deal with the choice set generation issue is proposed. This idea, however, results in a model which is equivalent to MNL over the universal choice set of paths and hence exhibits the IIA property. Relaxing IIA is challenging because the resulting dynamic programming problems become difficult to deal with and they need to be solved many times in order to estimate the models. We proposed different recursive models that allow to relax the IIA in the case of nesting (Chapter 3), generalized MEV (Chapter 4), mixing models (Chapter 5) and the RRM decision rule (Chapter 6). In order to deal with the model estimation we proposed the DeC method that allows to speed up the estimation of RL based models, which opens the possibility to estimate mixed RL and RRM/CRUM models. We also developed the value iteration method with dynamic accuracy to estimate the NRL model, and the method of integrating correlation structures to simplify the estimation of the RNMEV. We estimated the proposed models on real data, and they all had better in in-sample and out-of-sample fit than the RL one. It is important to note that, among these models, some perform better than others in terms of out-of-sample fit, e.g., the RCNL and mixed RL models are better than the NRL, and the ERRM performs the best in our experiments. Of course, advantages come at a price, as the RCNL
and mixed RL models are more expensive to estimate, compared to the NRL and RL ones, and the parameter estimates given by the RRM models are not straightforward to interpret. In addition, although performing worse than other models (the NRL, RCNL, mixed RL), the RL model can be estimated quickly through the DeC method, which may be useful for computing traffic flows or simulating path choices.

Furthermore, we showed how the logit models can be tested using White’s information matrix test. The outcome of the test suggested that alternative model structures should be investigated, e.g. the NRL (Chapter 3), the RCNL (Chapter 4), mixed RL (Chapter 5), and RRM/CRUM (Chapter 6) models.

We proposed, in Chapter 8, a new approach to quickly estimate MEV models with network-based correlation structures and large choice sets. The idea is based on the observation that the route choice modeling idea presented in Chapter 3 can be used to formulate MEV models as dynamic discrete choice models on networks of correlation structures. These networks are different from transport networks since link utilities are defined based on the structural parameters of the MEV models, and there are multiple destinations for each observation. We then showed that it is possible to benefit from the dynamic programming technique to estimate large-scale MEV models in short computational time.

Optimization methods for estimating discrete choice models have been studied in Chapter 9. This work is motivated by the fact that most of the discrete choice models (including those presented in this thesis) are estimated by maximum likelihood, and it is important to have an efficient optimization algorithm that can converge in as few iterations as possible since the objective functions are costly to evaluate. Mixed logit (Bastin et al., 2010, Cirillo and Xu, 2010), path-based MNL (Frejinger et al., 2009) and RL-RRM (Chapter 6) models were used to assess the performance of the proposed algorithms, but we note that other models presented in this thesis have also been estimated by the algorithms proposed in this chapter.

In the following we summarize the contributions of each chapter in more detail.

We have proposed the NRL model based on a new dynamic discrete choice framework in Chapter 3. The model relaxes the IIA property and allows path utilities to be correlated according to the structure of the transport network. We have proposed an iterative method with dynamic accuracy to solve the Bellman’s equation and estimate the model with the NFXP algorithm. Moreover, we have derived the gradients of the log-likelihood function which can be computed by solving systems of linear equations. We have provided numerical results using real data. The parameter estimates are sensible and the NRL model has significantly better fit than the RL model. We have also provided a cross-validation study suggesting that the NRL models are better than the RL models.
In Chapter 4, we have proposed a generalized dynamic route choice model based on a dynamic
discrete choice framework with correlated random terms. The generalized model (also called the
recursive network MEV, RNMEV) can be used to flexibly capture the correlation in the network
since it allows the choice model at each stage to be any member of the network MEV model (e.g.
the nested, cross-nested logit models). The resulting dynamic programming problem becomes
expensive to solve, and we have proposed the method of integrating correlation structures to
simplify the Bellman’s equation. This approach allowed us to use the value iteration method
from Chapter 3 to quickly estimate the RNMEV model on real data. We have applied the
proposed method to estimate a member of the RNMEV model with a cross-nesting correlation
structure at each choice stage. We have provided estimation results and performed a cross-
validation study using real data. The parameter estimates are sensible and the RCNL models
has significantly better fit and prediction than the other recursive models.

The estimation of the RL model is taken into account in more detail in Chapter 5. This
estimation requires solving several systems of linear equations, which makes its extensions, for
instance mixed RL models, impractical to estimate on real data. We have proposed the DeC
method that allows to speed up the estimation of the RL by reducing the number of systems to
be solved. We have applied the new method to quickly estimate two mixed RL models which use
random parameters and EC using subnetwork components. Based on a real data set, we have
provided estimation and predictions results. The results indicated that the mixed RL models
perform significantly better than the RL in fit and prediction. Moreover, we have showed that
the DeC allows to significantly speed up the estimation of the RL models (up to 30 times),
compared to the original approach.

In Chapter 6, we have considered the RRM decision rule proposed by Chorus (2010) and com-
pared the estimation results and prediction performance between the RRM and RUM frame-
works using dynamic route choice models. We have adapted the model proposed by Chorus
(2014) and we have proposed two variants: ERRM and ARRM. Moreover, in order to have
a fair comparison, we proposed models which are based on the RUM framework but have
utility functions that are similar to the RRM models. We showed that the RL models with
RRM and CRUM principles can be estimated on real data using the DeC method proposed in
Chapter 5. We provided numerical results and a cross-validation study using real data. The
cross-validation results indicated that the ERRM and ARRM models perform better than other
RRM- and RUM-based models, which suggest that it is important to flexibly capture the impact
of the non-chosen alternatives in the link regret functions.

In Chapter 7, we have shown that the information matrix test proposed by White (1982) can be
used to test path-based MNL models under sampling of alternatives (Frejinger et al., 2009) and
the RL models. The test statistic contains third derivatives of the log-likelihood function and we
addressed this issue by deriving the analytical Hessian for linear-in-parameters utility functions.
of the considered models, so that the third derivatives can be computed by finite difference. We have presented a Monte Carlo experiment based on real data to assess the size and power of the information matrix test. Moreover, we have compared the information matrix test results with those of the IIA test (Hausman and McFadden, 1984) and then the McTr test (McFadden and Train, 2000). We have also tested the models estimated on real observations showing that the null-hypotheses of the information matrix, IIA and McTr tests are strongly rejected for all the models. The results indicated that the link-size and path size attributes, that are designed to correct path utilities for correlation, significantly improve model fit but do not influence the outcome of these tests.

A new estimation technique for large-scale MEV models was proposed in Chapter 8. We proposed an approach based on a dynamic discrete choice framework to construct a group of MEV models with network-based correlation structures, i.e., the network MEV (Daly and Bierlaire, 2006). We showed that, under certain conditions, the dynamic models is consistent with McFadden’s MEV theory and generalizes the network MEV model. We presented a new estimation method that is convenient to use for estimating MEV models with large choice sets. More precisely, we used the concept of network flows to compute the choice probabilities, and estimated the model using the NFXP algorithm, and computed the model derivatives and elasticities via systems of linear equations. We presented numerical experiments using simulated data with different sizes of choice sets. The results indicated that we are able to quickly estimate the cross-nested and three-levels network MEV models with large choice sets. We have also provided a discussion on the computational time of the dynamic programming approach compared to recursive operations to show why our approach is convenient in the context.

In Chapter 9 we have revisited structured quasi-Newton techniques for maximum likelihood estimation. We focused on the impact of Hessian approximation methods on the performance of the trust region and line search algorithms, and explored the possibility of switching between various Hessian approximations throughout the optimization process. We proposed the predictive approach which does not require any new computation of the objective function, and the multi-subproblems model which requires additional evaluations of the objective function depending on the size of the set of Hessian approximations considered at each iteration. We have applied our methods to mixed logit and logit based route choice models, and our experiments clearly indicated the advantages of our approaches, compared to the others available in the literature.

In summary, this thesis presented several novel models and methods that allow to flexibly capture the correlation, quickly estimate and test large-scale discrete choice models in general, and route choice models in particular. Moreover, the proposed models and methods are implemented in MATLAB, and we share them freely as open source projects.
10.2 Future Research

In the following we present future work, starting with five ongoing research projects that have not been presented in the thesis. The first three ongoing projects are related to models and methods for static and deterministic networks. In the fourth project we apply our models for dynamic but deterministic networks. The last ongoing work concerns a recursive model and estimation methods for stochastic time-dependent networks.

In collaboration with Mælle Zimmermann and Mike Florian (INRO software), we have successfully estimated the NRL model (Chapter 3) and applied the DeC method (Chapter 5) for a bike route choice problem using GPS observations of cyclists trajectories in the city of Eugene, Oregon. The data was collected by the Central Lane Metropolitan Planning Organization. In this work, we show that the DeC method can be used to estimate a RL model of 14 parameters in approximately 1 hour on a network of 41,372 links and 118,959 connected link pairs, using a data set of 648 bike trajectories. We provide numerical results based on the RL and NRL models. We also focus our work on two applications of the recursive models: (i) traffic assignment and (ii) bike accessibility. More precisely, we show that the RL model possesses properties that enable both fast and accurate predictions, compared to models based on generated path choice sets.

The recursive route choice models presented in this thesis have some important differences compared to a general dynamic discrete choice models: (i) the Markov transition property are degenerate and (ii) there is no discounting of future utilities. The second difference implies that the proofs of Rust (1987) regarding the convergence of the value iteration methods do not apply. In an ongoing project we establish conditions for the existence of a fixed point solution and for the convergence of the value iteration method. We also derive Newton-Kantorovich (NK) iterations (Chapter 2, Section 2.2) and develop a switching strategy that allow the value iteration to converge rapidly to the fixed point solution. We provide numerical results based on private car and bike route choice data sets showing that our switching approach outperforms other classical ones in terms of computational time. This work therefore makes important contributions to the estimation of the recursive models because given a vector of parameters, fixed point solutions are required to compute the log-likelihood function, and we need to solve numerous value functions when searching over the parameter space.

All the route choice models presented in this thesis are based on dynamic discrete choice models without discount factor (the discount factor equals one). Estimating discount factors in dynamic models is an interesting way to capture additional information about the forward-looking behavior of the travelers. We have estimated the RL and NRL models with discount factors, where the discount factors vary according to subnetwork components. Our results indicate that the models with discount factors perform significantly better than those without in terms of
in-sample and out-of-sample fit. This work is still ongoing and we plan to estimate the models using several different data sets to confirm the findings.

Ramos et al. (2012) estimated a RL model for dynamic and deterministic networks, where a state is defined as time and location in these networks. One of our ongoing projects is to estimate models on this data that allow path utilities to be correlated, e.g. the NRL or RCNL models proposed in Chapters 3 and 4. These models can be directly applied but the estimation is challenging due to a large state space. This makes the corresponding dynamic programming problems costly to solve and we will therefore use the DeC method (Chapter 5) and the NK iterations (instead of the value iteration, see for instance Chapter 2, Section 2.2).

We are working on routing choice policy problems in stochastic time-dependent networks (Gao and Chabini, 2006), in collaboration with Song Gao, Jing Ding and Xinlian Yu (University of Massachusetts, Amherst). These problems are considerably more expensive to solve, compared to those considered in this thesis, due to the large number of states. In this context, a state is defined by time, location and perceived real-time information (due to day to day travel time variability). Moreover, the transition probability function is no longer degenerate. Therefore, the dynamic programing problems become very expensive to solve. We have designed new algorithms that allow to estimate a RL model on stochastic time-dependent networks in reasonable computational time. We have estimated that RL model using a data set of 500 observations (the study time period is from 6h30 to 9h AM) on a subnetwork of Stockholm (Sweden) which consists of 2772 nodes and 5447 links (including 619 stochastic links) (Yu et al., 2015). We are currently working on estimating different methods to measure the travel time variability as well as on improving the performance of the estimation methods.

In the following we present different research topics that we would like to investigate in the future.

The network-based models that are proposed in this thesis are flexible in the sense that they can be used for many applications where decisions can be represented by a network. An interesting direction for future work is to extend the models to other network settings than the ones already covered. For example, route choice in multi-modal and/or activity networks and combine different choices such as departure time, route and destination choice.

As we have discussed at length, the key challenge associated with dynamic discrete choice models is solving the corresponding dynamic programming problems. The DeC method proposed in Chapter 5 is designed for estimating the RL model, and we have shown that it can speed up the estimation up to 30 times. In future work, we plan to adapt this method so that it can be used with other models, e.g., NRL, RNMEV models proposed in Chapters 3, 4, respectively. This can make our dynamic models more attractive in practice, especially if they are integrated in simulation models.
We have proposed the switching methods to improve the performance of the optimization algorithms for the maximum likelihood estimation. Other switching strategies could be investigated, e.g., at each iteration the values of the objective function from previous iterations can be taken into account to improve the forecasting of the next Hessian approximation matrix. Moreover, it would be interesting to apply the switching approaches for other problems, e.g., least-squares. The convergence properties of the switching approaches under the trust region and line search frameworks could also be interesting to investigate.

The estimation techniques that we have proposed could be used in other dynamic discrete choice applications. Since estimating dynamic discrete choice models with correlated random terms is a topic of interest (for instance Aguirregabiria and Mira, 2010), we would like to use the method presented Chapter 4 with the NFXP algorithm to estimate dynamic discrete choice models with MEV random terms. The DeC method proposed in Chapter 5 can also be adapted to speed up the estimation in other dynamic discrete choice applications. We are also interested in the use of Bayesian procedures and the Metropolis-Hasting algorithm (see, for instance, Train, 2003) for estimating mixed recursive models.

For more long term research activities, it would be interesting to investigate the integration of the recursive models proposed here and optimization models, e.g., the maximum capture problems (for instance Benati and Hansen, 2002). Another interesting direction would be to incorporate dynamic models of strategic customer behavior in revenue management problems (Talluri and Van Ryzin, 2004).
Appendix A

Supplement to Chapter 4

A.1 Proof of Theorem 4.2

Consider the network of correlation structure $G_k = (S_k, A_k, C_k)$ at state $k$. In order to prove the result we derive the function $G_k(e(v(a))(Y^*_a)^{1/\mu_a}, a \in S(k))$ using (4.6) and (4.7), and note that $G_k(y) = G^*_k(y)$, where $r$ is the root of $G_k$. For notational simplicity we denote $y^*_k$ as a vector of size $|S_k|$ with entries $e(v(a))(Y^*_a)^{1/\mu_a}$, for all $a \in S(k)$. We also note that, as discussed in Section 4.3, state $k$ is also the root $r$ and the choice set $C_k$ is identical to $S(k)$.

We first introduce some definitions. For each state $i \in S_k$, we denote $L_k(i)$ as the length (defined as number of arcs) of the longest sequences of states (or paths) connecting $i$ and all $j \in S(k)$ via states in $S_k$. $L_k(i)$ is finite since the network $G_k$ is cycle-free. For any integer number $p \geq 0$ we denote $T^k(p)$ as the set of state $i$ such that $i \in S_k$ and $L_k(i) = p$. In other words

$$T^k(p) = \{i|i \in S_k, L_k(i) = p\}, \forall p \in \mathbb{N}.$$

We have the following proposition, which is easy to verify

**Proposition A.1.**

(i) $T^k(0) = S(k), \bigcup_{t=1}^{L^k(r)} = S_k$.

(ii) $T^k(p) \cap T^k(q) = \emptyset, p, q \geq 0, p \neq q$.

(iii) Given state $i \in T^k(p), p \geq 1$, if $j \in S^*(i)$ then $j \in \bigcup_{t=0}^{p-1} T^k(t)$.

**Proof.** (i) and (ii) are trivial to verify. For (iii), we suppose that $j \notin \bigcup_{t=0}^{p-1} T^k(t)$, then there exits a number $p' \geq p$ such that $j \in T^k(p')$. It means that there exits a sequence of length $p'$ connecting $j$ and states in $S(k)$. Moreover, since $j \in S^*(i)$ and from the fact that $i \notin S(k)$ (because $i \in T^k(p)$ and $p \geq 1$) we have $j \in S_k(i)$ due to Proposition 4.1(iv). Consequently, there
exits a sequence of length \( p' + 1 > p \) connecting \( i \) and states in \( S(k) \). This is in contradiction with the assumption that \( i \in T^k(p) \). So \( j \) has to be in \( \bigcup_{t=0}^{p-1} T^k(t) \) and (iii) is proved. \( \square \)

For all \( i \in S_k \), the values of \( G^i_k(y_k^*) \) can be computed based on (4.6) and (4.7) as

\[
G^i_k(y_k^*) = \left( e^{v(i)(Y_i^*)^{1/\mu_i}} \right)^{\xi^k_i}, \forall i \in S(k). \tag{A.1}
\]

For each \( i \in S_k \setminus S(k) \) we have

\[
G^i_k(y_k^*) = \sum_{j \in S_k(i)} \alpha^i_k G^j_k(y_k^*)^{\xi^k_j / \xi^k_i}. \tag{A.2}
\]

We introduce the following lemma

**Lemma A.2.** Given state \( k \in S \), if \( G^i_k(y_k^*) \), \( \forall i \in S_k \), are computed based on (A.1) and (A.2) then

\[
G^i_k(y_k^*) = Y_i^*, \forall i \in T^k(p), \forall p \in \mathbb{Z}^+. \tag{A.3}
\]

**Proof.** Based on the definitions in (4.10) and (4.11), (A.1) can be written equivalently as

\[
G^i_k(y_k^*) = e^{\xi^k_i v(i)(Y_i^*)^{1/\mu_i}} / \mu_i, \forall i \in S(k). \tag{A.4}
\]

Here we remark that \( \xi^k_i \neq \mu_i^*, \forall i \in S(k), \) and \( \xi^k_i = \mu_i^* \forall i \in \mathbb{S}_k \setminus S(k) \). Due to Proposition 4.1(iv) we have \( S_k(i) = S^*(i), \forall i \in \mathbb{S}_k \setminus S(k), \) so (A.2) can be written as

\[
G^i_k(y_k^*) = \sum_{j \in S^*(i) \setminus S(k)} \sum_{j \in S(k)} e^{\mu_i^* v^*(j|i)} \left( G^j_k(y_k^*) \right)^{\mu_j^* / \mu_i^*} \alpha^i_k G^j_k(y_k^*)^{\xi^k_j / \xi^k_i} + \sum_{j \in S^*(i) \setminus S(k)} e^{\mu_i^* v^*(j|i) - v(j|j)} \left( G^j_k(y_k^*) \right)^{\mu_j^* / \xi^k_i}. \tag{A.5}
\]

\[ \forall i \in \mathbb{S}_k \setminus S(k). \]

Substituting (A.4) into (A.5) we obtain

\[
G^i_k(y_k^*) = \sum_{j \in S^*(i) \setminus S(k)} e^{\mu_i^* v^*(j|i)} \left( G^j_k(y_k^*) \right)^{\mu_j^* / \mu_i^*} + \sum_{j \in S^*(i) \setminus S(k)} e^{\mu_i^* v^*(j|i) - v(j|j)} \left( G^j_k(y_k^*) \right)^{\mu_j^* / \xi^k_i}, \forall i \in \mathbb{S}_k \setminus S(k). \tag{A.6}
\]

Now we prove the result by induction. For \( p = 1 \), according to Proposition A.1(i)-(iii) we have the fact that for each \( i \in T^k(1) \), if \( j \in S^*(i) \) then \( j \in T^k(0) \), or equivalently \( j \in S(k) \). Thus, (A.6) can be written as

\[
G^i_k(y_k^*) = \sum_{j \in S^*(i)} e^{\mu_i^* v^*(j|i)} \left( Y_j^* \right)^{\mu_j^* / \mu_i^*}, \forall i \in T^k(1). \tag{A.7}
\]
So from (4.15) and (A.7) we have \( G_k^i(y_k^i) = Y_i^* \forall i \in T^k(1) \), meaning that (A.3) is true for \( p = 1 \). Now we assume that the result is true for \( p \geq 1 \). In other words

\[
G_k^i(y_k^i) = Y_i^*, \forall i \in \bigcup_{t=1}^p T^k(t).
\]

We will prove that the result is also true for \( p + 1 \). For each state \( i \in T^k(p+1) \), according to Proposition A.1(iv), if \( j \in S^+(i)\setminus S(k) \) then \( j \in \bigcup_{t=1}^p T^k(t) \). Consequently, by assumption,

\[
G_k^j(y_k^j) = Y_j^* \forall j \in S^+(i)\setminus S(k).
\]

Hence, (A.6) can be written as

\[
G_k^i(y_k^i) = \sum_{j \in S^+(i)} e^{\mu_i^*v^*(j|i)} (Y_j^*)^{\mu_i^*/\mu_j^*} + \sum_{j \in S^+(i)\setminus S(k)} e^{\mu_i^*v^*(j|i)} (Y_j^*)^{\mu_i^*/\mu_j^*}, \forall i \in T^k(p+1),
\]

so \( G_k^i(y_k^i) = Y_i^* \), \( \forall i \in T^k(p+1) \), because of (4.15). This validates (A.3) for \( p+1 \), as required.

We note that if \( p = L^k(k) \) then \( G_k^i(y_k^i) = Y_k^* \), or \( Y_k^* = G_k(y_k^i) \). Hence, Theorem 4.2 is proved.

### A.2 Proof of Theorem 4.3

We consider the network \( G_k = (S_k, A_k, C_k) \) at state \( k \in S \). Under the hypotheses of Theorem 4.3 we have the fact that \( y_k = y_k^i \) (we recall that \( y_k \) is a vector of size \(|S(k)|\) with elements \( e^{\nu(a|k)}(Y_a)^{1/\mu_a}, \forall a \in S(k) \)). So from (4.8), the choice probability \( P(a|k) \forall a \in S(k) \) given by the network MEV model at state \( k \) is

\[
P(a|k) = \sum_{[j_0, \ldots, j_{I-1}] \in \Omega^k(a)} \prod_{t=0}^{I-1} \frac{G_k^{j_{t+1}}(y_k^i)^{\ell_{j_{t+1}}/\ell_{j_t}}}{G_k^i(y_k^i)}. \tag{A.9}
\]

Given two states \( i, j \in S_k \), \( j \in S^+(i) \), we consider two cases: \( j \notin S(k) \) or \( j \in S(k) \).

- If \( j \notin S(k) \), then according to Lemma A.2, we have \( G_k^i(y_k^i) = Y_i^* \) and \( G_k^j(y_k^j) = Y_j^* \). Furthermore, from the definitions in (4.10), (4.11) and Equation 4.16 we obtain

\[
\frac{\alpha_k^j(G_k^j(y_k^j))^{\ell_j/\ell_k}}{G_k^j(y_k^j)} = e^{\mu^*_jv^*(j|i)} (Y_j^*)^{\mu_j^*/\mu_j^*} = P^i(j|i). \tag{A.10}
\]
• If \( j \in S(k) \), then from (A.4) we have

\[
\frac{\alpha_{ij}^k(G^j_k(y^*_k))^{\xi^j_k/\xi^i_j}}{G^i_k(y^*_k)} = \frac{e^{\mu^*_i(v^*(j|i) - v(j|k))} Y^*_i}{Y^*_j} \frac{e^{\mu^*_j v(j|k)} (Y^*_j)^{\mu^*_j/\mu^*_i}}{Y^*_i} = P^*(j|i).
\]

(A.11)

So from (A.10) and (A.11) we obtain

\[
\frac{\alpha_{ij}^k(G^j_k(y^*_k))^{\xi^j_k/\xi^i_j}}{G^i_k(y^*_k)} = P^*(j|i), \quad \forall i \in S \setminus S(k), j \in S^*(i),
\]

and the choice probability \( P(a|k) \) in (A.9) can be computed as

\[
P(a|k) = \sum_{[j_0, \ldots, j_l] \in \Omega^k(a)} \prod_{t=0}^{l-1} P^*(j_{t+1}|j_t).
\]
Appendix B

Supplement to Chapter 5
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob. Std. Err.</td>
<td>0.106</td>
<td>0.107</td>
<td>0.108</td>
<td>0.113</td>
<td>0.114</td>
<td>0.101</td>
<td>0.112</td>
<td>0.114</td>
<td>0.100</td>
<td>0.113</td>
</tr>
<tr>
<td>(\sigma_{TT})</td>
<td>0.800</td>
<td>0.688</td>
<td>0.697</td>
<td>-1.058</td>
<td>0.811</td>
<td>0.606</td>
<td>0.817</td>
<td>0.858</td>
<td>0.845</td>
<td>-0.810</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.032</td>
<td>0.027</td>
<td>0.028</td>
<td>0.047</td>
<td>0.033</td>
<td>0.024</td>
<td>0.034</td>
<td>0.036</td>
<td>0.022</td>
<td>0.033</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>25.02</td>
<td>25.19</td>
<td>24.49</td>
<td>-22.42</td>
<td>24.35</td>
<td>25.31</td>
<td>24.33</td>
<td>24.05</td>
<td>38.41</td>
<td>-24.20</td>
</tr>
<tr>
<td>(\hat{\beta}_{LT})</td>
<td>-0.928</td>
<td>-0.928</td>
<td>-0.927</td>
<td>-0.926</td>
<td>-0.927</td>
<td>-0.926</td>
<td>-0.926</td>
<td>-0.926</td>
<td>-0.926</td>
<td>-0.927</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-30.18</td>
<td>-30.30</td>
<td>-30.09</td>
<td>-29.79</td>
<td>-29.99</td>
<td>-30.39</td>
<td>-29.96</td>
<td>-29.91</td>
<td>-30.41</td>
<td>-30.01</td>
</tr>
<tr>
<td>(\hat{\beta}_{LC})</td>
<td>-0.425</td>
<td>-0.424</td>
<td>-0.426</td>
<td>-0.430</td>
<td>-0.427</td>
<td>-0.423</td>
<td>-0.428</td>
<td>-0.423</td>
<td>-0.423</td>
<td>-0.427</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.012</td>
<td>0.012</td>
<td>0.013</td>
<td>0.013</td>
<td>0.013</td>
<td>0.012</td>
<td>0.013</td>
<td>0.013</td>
<td>0.012</td>
<td>0.013</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-34.02</td>
<td>-33.93</td>
<td>-33.91</td>
<td>-33.70</td>
<td>-33.93</td>
<td>-33.95</td>
<td>-33.89</td>
<td>-33.93</td>
<td>-33.93</td>
<td>-33.92</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
<td>0.115</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-38.31</td>
<td>-38.39</td>
<td>-38.26</td>
<td>-38.04</td>
<td>-38.19</td>
<td>-38.44</td>
<td>-38.19</td>
<td>-38.15</td>
<td>-38.46</td>
<td>-38.18</td>
</tr>
<tr>
<td>Final LL</td>
<td>-6233.45</td>
<td>-6239.26</td>
<td>-6229.30</td>
<td>-6213.25</td>
<td>-6223.86</td>
<td>-6244.47</td>
<td>-6223.50</td>
<td>-6220.64</td>
<td>-6245.80</td>
<td>-6223.35</td>
</tr>
</tbody>
</table>

Table B.1: Estimation results for the mixed RL with random parameters
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob. Std. Err.</td>
<td>0.118</td>
<td>0.119</td>
<td>0.118</td>
<td>0.119</td>
<td>0.120</td>
<td>0.118</td>
<td>0.119</td>
<td>0.119</td>
<td>0.120</td>
<td>0.117</td>
</tr>
<tr>
<td>$\hat{\beta}_{LT}$</td>
<td>-0.947</td>
<td>-0.945</td>
<td>-0.946</td>
<td>-0.949</td>
<td>-0.942</td>
<td>-0.946</td>
<td>-0.947</td>
<td>-0.941</td>
<td>-0.949</td>
<td>-0.943</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
<td>0.031</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-30.45</td>
<td>-30.42</td>
<td>-30.50</td>
<td>-30.70</td>
<td>-30.36</td>
<td>-30.56</td>
<td>-30.46</td>
<td>-30.58</td>
<td>-30.57</td>
<td>-30.46</td>
</tr>
<tr>
<td>$\hat{\beta}_{LC}$</td>
<td>-0.406</td>
<td>-0.406</td>
<td>-0.406</td>
<td>-0.403</td>
<td>-0.403</td>
<td>-0.405</td>
<td>-0.405</td>
<td>-0.404</td>
<td>-0.403</td>
<td>-0.405</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.116</td>
<td>0.117</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.117</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-37.74</td>
<td>-37.64</td>
<td>-37.68</td>
<td>-37.75</td>
<td>-37.74</td>
<td>-37.71</td>
<td>-37.73</td>
<td>-37.78</td>
<td>-37.76</td>
<td>-37.53</td>
</tr>
<tr>
<td>$\sigma_{RDS}$</td>
<td>1.700</td>
<td>1.893</td>
<td>1.630</td>
<td>1.736</td>
<td>2.055</td>
<td>1.829</td>
<td>1.756</td>
<td>1.712</td>
<td>2.009</td>
<td>1.601</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.045</td>
<td>0.045</td>
<td>0.033</td>
<td>0.038</td>
<td>0.072</td>
<td>0.038</td>
<td>0.041</td>
<td>0.042</td>
<td>0.077</td>
<td>0.035</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>37.62</td>
<td>42.41</td>
<td>49.20</td>
<td>45.31</td>
<td>28.53</td>
<td>48.10</td>
<td>42.54</td>
<td>41.05</td>
<td>26.09</td>
<td>45.11</td>
</tr>
<tr>
<td>$\sigma_{RDS}$</td>
<td>1.796</td>
<td>1.553</td>
<td>1.663</td>
<td>1.661</td>
<td>1.482</td>
<td>1.487</td>
<td>1.781</td>
<td>1.942</td>
<td>1.702</td>
<td>1.892</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.155</td>
<td>0.079</td>
<td>0.036</td>
<td>0.074</td>
<td>0.145</td>
<td>0.054</td>
<td>0.189</td>
<td>0.408</td>
<td>0.088</td>
<td>0.258</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>11.61</td>
<td>19.71</td>
<td>46.19</td>
<td>22.57</td>
<td>10.25</td>
<td>27.49</td>
<td>9.41</td>
<td>4.75</td>
<td>19.23</td>
<td>7.32</td>
</tr>
<tr>
<td>$\sigma_{RDS}$</td>
<td>1.789</td>
<td>1.636</td>
<td>1.472</td>
<td>1.671</td>
<td>1.420</td>
<td>1.514</td>
<td>1.718</td>
<td>1.645</td>
<td>1.816</td>
<td>1.446</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.074</td>
<td>0.076</td>
<td>0.033</td>
<td>0.077</td>
<td>0.044</td>
<td>0.034</td>
<td>0.077</td>
<td>0.034</td>
<td>0.111</td>
<td>0.043</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>24.19</td>
<td>21.58</td>
<td>45.19</td>
<td>21.64</td>
<td>32.27</td>
<td>45.16</td>
<td>22.24</td>
<td>48.28</td>
<td>16.36</td>
<td>33.59</td>
</tr>
<tr>
<td>$\sigma_{RDS}$</td>
<td>1.567</td>
<td>1.802</td>
<td>1.881</td>
<td>1.543</td>
<td>1.701</td>
<td>1.433</td>
<td>1.759</td>
<td>1.525</td>
<td>1.338</td>
<td>1.614</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.035</td>
<td>0.040</td>
<td>0.042</td>
<td>0.035</td>
<td>0.038</td>
<td>0.032</td>
<td>0.040</td>
<td>0.034</td>
<td>0.030</td>
<td>0.036</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>44.47</td>
<td>44.58</td>
<td>45.02</td>
<td>44.90</td>
<td>44.30</td>
<td>44.83</td>
<td>44.63</td>
<td>44.10</td>
<td>45.01</td>
<td></td>
</tr>
<tr>
<td>$\sigma_{RC}$</td>
<td>2.464</td>
<td>2.869</td>
<td>2.634</td>
<td>2.825</td>
<td>2.855</td>
<td>2.762</td>
<td>2.525</td>
<td>2.692</td>
<td>2.762</td>
<td>2.513</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.051</td>
<td>0.161</td>
<td>0.237</td>
<td>0.081</td>
<td>0.138</td>
<td>0.149</td>
<td>0.045</td>
<td>0.144</td>
<td>0.099</td>
<td>0.037</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>48.09</td>
<td>17.82</td>
<td>11.16</td>
<td>34.82</td>
<td>20.76</td>
<td>18.51</td>
<td>55.53</td>
<td>18.76</td>
<td>27.78</td>
<td>67.82</td>
</tr>
</tbody>
</table>

**Final LL** | -5971.71 | -5962.38 | -5945.92 | -5977.19 | -5961.04 | -5969.28 | -5956.51 | -5989.93 | -5987.23 | -5981.71 |

**Table B.2:** Estimation results for the mixed RL model with EC
### Table B.3: Estimation results for the mixed RL model with EC and RP

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob. Std. Err.</td>
<td>0.134</td>
<td>0.125</td>
<td>0.134</td>
<td>0.123</td>
<td>0.121</td>
<td>0.131</td>
<td>0.122</td>
<td>0.122</td>
<td>0.133</td>
</tr>
<tr>
<td>( \sigma_{TT} )</td>
<td>0.886</td>
<td>0.604</td>
<td>0.910</td>
<td>0.755</td>
<td>0.657</td>
<td>1.106</td>
<td>1.052</td>
<td>0.932</td>
<td>0.875</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.079</td>
<td>0.109</td>
<td>0.041</td>
<td>0.029</td>
<td>0.025</td>
<td>0.046</td>
<td>0.045</td>
<td>0.037</td>
<td>0.034</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>11.161</td>
<td>5.525</td>
<td>22.328</td>
<td>25.878</td>
<td>26.605</td>
<td>24.114</td>
<td>23.175</td>
<td>24.908</td>
<td>25.618</td>
</tr>
<tr>
<td>( \beta_{LT} )</td>
<td>-0.941</td>
<td>-0.943</td>
<td>-0.933</td>
<td>-0.936</td>
<td>-0.945</td>
<td>-0.939</td>
<td>-0.937</td>
<td>-0.931</td>
<td>-0.947</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.031</td>
<td>0.031</td>
<td>0.032</td>
<td>0.032</td>
<td>0.032</td>
<td>0.032</td>
<td>0.032</td>
<td>0.032</td>
<td>0.032</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>-30.144</td>
<td>-29.978</td>
<td>-29.351</td>
<td>-29.613</td>
<td>-29.417</td>
<td>-29.232</td>
<td>-29.375</td>
<td>-29.799</td>
<td>-30.634</td>
</tr>
<tr>
<td>( \beta_{LC} )</td>
<td>-0.413</td>
<td>-0.406</td>
<td>-0.422</td>
<td>-0.411</td>
<td>-0.416</td>
<td>-0.417</td>
<td>-0.420</td>
<td>-0.417</td>
<td>-0.418</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.015</td>
<td>0.015</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
<td>0.014</td>
</tr>
<tr>
<td>( \beta_{UT} )</td>
<td>-4.320</td>
<td>-4.370</td>
<td>-4.301</td>
<td>-4.348</td>
<td>-4.328</td>
<td>-4.319</td>
<td>-4.331</td>
<td>-4.318</td>
<td>-4.310</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.115</td>
<td>0.117</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
<td>0.116</td>
</tr>
<tr>
<td>( \sigma_{RSON} )</td>
<td>1.787</td>
<td>2.084</td>
<td>1.744</td>
<td>1.933</td>
<td>1.655</td>
<td>2.100</td>
<td>1.976</td>
<td>2.221</td>
<td>1.771</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.107</td>
<td>0.134</td>
<td>0.055</td>
<td>0.053</td>
<td>0.033</td>
<td>0.059</td>
<td>0.037</td>
<td>0.059</td>
<td>0.039</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>16.673</td>
<td>15.543</td>
<td>31.949</td>
<td>36.286</td>
<td>50.902</td>
<td>35.573</td>
<td>53.667</td>
<td>37.798</td>
<td>44.850</td>
</tr>
<tr>
<td>( \sigma_{RSOS} )</td>
<td>1.582</td>
<td>1.590</td>
<td>1.331</td>
<td>1.644</td>
<td>1.574</td>
<td>1.332</td>
<td>1.832</td>
<td>1.227</td>
<td>1.681</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.228</td>
<td>0.122</td>
<td>0.046</td>
<td>0.267</td>
<td>0.215</td>
<td>0.093</td>
<td>0.228</td>
<td>0.056</td>
<td>0.253</td>
</tr>
<tr>
<td>( \sigma_{RSON} )</td>
<td>1.629</td>
<td>1.898</td>
<td>1.579</td>
<td>1.755</td>
<td>1.721</td>
<td>1.624</td>
<td>1.644</td>
<td>1.697</td>
<td>1.528</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.030</td>
<td>0.096</td>
<td>0.049</td>
<td>0.020</td>
<td>0.101</td>
<td>0.031</td>
<td>0.056</td>
<td>0.043</td>
<td>0.036</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>54.30</td>
<td>19.778</td>
<td>31.998</td>
<td>87.75</td>
<td>17.084</td>
<td>52.399</td>
<td>29.521</td>
<td>39.173</td>
<td>42.637</td>
</tr>
<tr>
<td>( \sigma_{RSOS} )</td>
<td>1.777</td>
<td>1.326</td>
<td>1.693</td>
<td>1.710</td>
<td>1.725</td>
<td>1.356</td>
<td>1.489</td>
<td>1.501</td>
<td>1.696</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.059</td>
<td>0.100</td>
<td>0.043</td>
<td>0.033</td>
<td>0.037</td>
<td>0.029</td>
<td>0.030</td>
<td>0.030</td>
<td>0.039</td>
</tr>
<tr>
<td>Rob. t-test(0)</td>
<td>29.934</td>
<td>13.193</td>
<td>38.987</td>
<td>51.974</td>
<td>46.881</td>
<td>46.479</td>
<td>49.535</td>
<td>49.761</td>
<td>43.927</td>
</tr>
<tr>
<td>( \sigma_{RC} )</td>
<td>2.833</td>
<td>2.928</td>
<td>2.763</td>
<td>3.080</td>
<td>2.843</td>
<td>2.977</td>
<td>2.645</td>
<td>2.702</td>
<td>3.012</td>
</tr>
<tr>
<td>Rob. Std. Err.</td>
<td>0.037</td>
<td>0.094</td>
<td>0.124</td>
<td>0.269</td>
<td>0.092</td>
<td>0.178</td>
<td>0.063</td>
<td>0.062</td>
<td>0.231</td>
</tr>
<tr>
<td>Final LL</td>
<td>-5895.62</td>
<td>-5955.43</td>
<td>-5888.26</td>
<td>-5922.38</td>
<td>-5907.29</td>
<td>-5904.32</td>
<td>-5937.29</td>
<td>-5911.62</td>
<td>-5900.71</td>
</tr>
</tbody>
</table>
Appendix C

Supplement to Chapter 7

C.1 A Proof for the Validity of the Information Matrix Equality when Choice Sets are Sampled

This appendix gives a proof for the validity of the information matrix equality for the path-based model with sampled choice sets when the model is correctly specified and the sampling corrections are added to the choice probabilities.

As in Section 7.3, we denote the probability of sampling a set of alternatives \( D \), given observed choice \( \sigma \), by \( \pi(D|\sigma) \). If we assume that the chosen path is always included in \( C \), i.e. \( \pi(C|j) = 0 \) if \( j \notin C \), the unconditional probability to select the sample \( C \) is

\[
\pi(C) = \sum_{j \in \Omega} P(j) \pi(C|j) = \sum_{j \in C} P(j) \pi(C|j). 
\]  

(C.1)

The unconditional probability over choices and alternative samples of \( \ln P(\sigma, \beta|C) \) is

\[
\sum_{\sigma \in \Omega} \sum_{C \subseteq \Omega \atop C \ni \sigma} P(\sigma) \pi(C|\sigma) \ln P(\sigma, \beta|C),
\]

where \( P(\sigma) \) is the true choice probability of \( \sigma \) and the sum can be limited to the sets \( C \) satisfying \( \pi(C|\sigma) > 0 \). Replacing \( P(\sigma) \) by the logit expression, we have

\[
\sum_{\sigma \in \Omega} \sum_{C \subseteq \Omega \atop C \ni \sigma} \frac{\pi(C|\sigma)e^{\frac{1}{\mu}v(x_{\sigma n}, \beta^*)}}{\sum_{j \in \Omega} e^{\frac{1}{\mu}v(x_{jn}, \beta^*)}} \frac{\pi(C|j)e^{\frac{1}{\mu}v(x_{jn}, \beta^*)}}{\sum_{j \in C} \pi(C|j)e^{\frac{1}{\mu}v(x_{jn}, \beta^*)}} \ln P(\sigma, \beta|C).
\]
By (7.6) and (C.1), this is equivalent to
\[
\sum_{C \subseteq \Omega} \sum_{\sigma \in C} P(\sigma|C) \pi(C) \ln P(\sigma, \beta|C). \tag{C.2}
\]

The expectation over the population \( Y \) of the Hessian of (C.2) over \( \beta \) is
\[
H(\beta) = \mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma|C) \nabla^2_{\beta\beta} \ln P(\sigma, \beta|C) \right],
\]
and the expected outer product of the gradient is
\[
I(\beta) = \mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma|C) \nabla_\beta \ln P(\sigma, \beta|C) \nabla_\beta \ln P(\sigma, \beta|C)^T \right].
\]

We now show that \( H(\beta^*) + I(\beta^*) = 0 \). Since, for any \( \beta \), \( \sum_{\sigma \in C} P(\sigma, \beta|C) = 1 \), we have
\[
\sum_{\sigma \in C} \nabla_\beta P(\sigma, \beta|C) = \sum_{\sigma \in C} P(\sigma, \beta|C) \nabla_\beta \ln P(\sigma, \beta|C) = 0,
\]
and
\[
\mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma, \beta|C) \nabla_\beta \ln P(\sigma, \beta|C) \right] = 0.
\]

Let assume that some regularity conditions allowing to permute the expectation and the derivative operators are satisfied. We then have
\[
\mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma, \beta|C) \nabla^2_{\beta\beta} \ln P(\sigma, \beta|C)
\right.
\]
\[
+ \pi(C) \nabla_\beta P(\sigma, \beta|C) \nabla_\beta \ln P(\sigma, \beta|C)^T \right] = 0. \tag{C.3}
\]

At \( \beta^* \), (C.3) becomes
\[
\mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma|C) \nabla^2_{\beta\beta} \ln P(\sigma|C) \right]
\]
\[
= -\mathbb{E}_Y \left[ \sum_{C \subseteq \Omega} \sum_{\sigma \in C} \pi(C) P(\sigma|C) \nabla_\beta \ln P(\sigma|C) \nabla_\beta \ln P(\sigma|C)^T \right],
\]
i.e. \( H(\beta^*) = -I(\beta^*) \), as announced. In other terms, the information matrix equality holds for sampled choice sets.
C.2 Derivatives of the Value Functions

In this appendix we derive the analytical formula for the first and second derivatives of the value functions. The first derivative of equation (7.8) with respect to one parameter $\beta_q$ is

$$\frac{\partial z^d(\beta)}{\partial \beta_p} = [I - M^d(\beta)]^{-1} \left[ \frac{\partial M^d(\beta)}{\partial \beta_p} \right] z^d(\beta)$$  \hspace{1cm} (C.4)

We denote the Jacobian as $\nabla_{\beta} z^d(\beta)$, and the second derivative of this equation with respect to two parameters $\beta_p, \beta_q$ is

$$\frac{\partial^2 z^d(\beta)}{\partial \beta_p \partial \beta_q} = \frac{\partial [I - M^d(\beta)]^{-1} \frac{\partial M^d(\beta)}{\partial \beta_p}}{\partial \beta_q} z^d(\beta) + [I - M^d(\beta)]^{-1} \frac{\partial^2 M^d(\beta)}{\partial \beta_p \partial \beta_q} z^d(\beta).$$  \hspace{1cm} (C.5)

Note that

$$\frac{\partial [I - M^d(\beta)]^{-1}}{\partial \beta_q} = [I - M^d(\beta)]^{-1} \frac{\partial M^d(\beta)}{\partial \beta_q} [I - M^d(\beta)]^{-1}$$

and using (C.4), we obtain a concise analytical expression of $\frac{\partial^2 z^d(\beta)}{\partial \beta_p \partial \beta_q}$

$$\frac{\partial^2 z^d(\beta)}{\partial \beta_p \partial \beta_q} = [I - M^d(\beta)]^{-1} \left( \frac{\partial^2 M^d(\beta)}{\partial \beta_p \partial \beta_q} z^d(\beta) + \frac{\partial M^d(\beta)}{\partial \beta_p} \frac{\partial z^d(\beta)}{\partial \beta_q} + \frac{\partial M^d(\beta)}{\partial \beta_q} \frac{\partial z^d(\beta)}{\partial \beta_p} \right),$$  \hspace{1cm} (C.6)

which is a linear system and is fairly fast to compute.
C.3 Detailed Numerical Results

This appendix provides the detailed cumulative distributions of $p$-values given by the Monte Carlo experiment.

**Figure C.1**: Cumulative distribution of $p$-values used to assess the number of type I errors, RL-LS model
Figure C.2: Cumulative distribution of p-values used to assess the number of type I errors, RL and PL models.
Figure C.3: Cumulative distribution of $p$-values used to assess the number of type II errors, RL and PL models
Appendix D

Supplement to Chapter 8

D.1 Proof of Theorem 8.2

This appendix presents a proof for Theorem 8.2. We first introduce some definitions.

**Definition D.1.** Given a set of nodes \( U \subseteq \mathcal{N} \) and \( k, a \in U \), we denote \( \Theta(k, a, U) \) be the set of paths of strictly positive lengths that connect \( k \) and \( a \) via nodes in \( U \), i.e., \( \Theta(k, a, U) = \{[k_0, \ldots, k_J]|k_0 = k; k_J = a; k_i \in U; k_{i+1} \in \mathcal{N}(k_i) \forall i = 0, \ldots, J - 1; J \geq 1\} \).

Note that if \( k = a \) and \( \Theta(k, a, U) \neq \emptyset \), then \( \Theta(k, a, U) \) contains all the cycles going from \( k \) and come back to \( k \) via nodes in \( U \).

**Definition D.2.** Given a set of nodes \( U \subseteq \mathcal{N} \) and \( k, a \in U \), \( \pi(k, a, U) = 1 \) if \( \Theta(k, a, U) \neq \emptyset \) or \( k \equiv a \), and \( \pi(k, a, U) = 0 \) otherwise.

**Definition D.3.** Given a set \( U \subseteq \mathcal{N} \) and a node \( k \in U \), \( K(k, U) \) is the set of nodes that are connected to \( k \) via nodes in \( U \), i.e., \( K(k, U) = \{a|\pi(k, a, U) = 1\} \), \( \forall k \in U \).

**Definition D.4.** Given a set \( U \subseteq \mathcal{N} \), we define \( \mathcal{Y}(U) \) to be a system of non-linear equations which contains the following equations

\[
Y_k = \begin{cases} 
\sum_{a \in \mathcal{N}(k) \cap U} M_{ka} Y_a^{\mu_k/\mu_a} & \text{if } k \in U \setminus C \\
e^{U_k} & \text{if } k \in U \cap C \\
0 & \text{if } k \notin C, \mathcal{N}(k) \cap U = \emptyset,
\end{cases}
\]  

(D.1)

where \( M_{ka}, \forall a, k \in \mathcal{N} \), are defined in (8.20).

**Lemma D.5.** Under the hypotheses in Theorem 8.2, given \( U \subseteq \mathcal{N} \) and two nodes \( a, k \in U \), if \( \pi(k, a, U) = 1 \) and \( \pi(a, k, U) = 1 \) then \( \mu_a = \mu_k \).
Proof. The lemma is obviously verified, as \( \mu_k \leq \mu_a \) if \( \pi(k, a, \mathcal{U}) = 1, \ \forall k, a \in \mathcal{U} \).

Lemma D.6. Under the hypotheses of Theorem 8.2 and given a set \( \mathcal{U} \subseteq \mathcal{N} \). If \( (Y_k, k \in \mathcal{U}) \) is a vector solution to \( \mathcal{Y}(\mathcal{U}) \), then given a node \( k_0 \in \mathcal{U} \setminus \mathcal{C} \) and \( \mathcal{N}(k_0) \cap \mathcal{U} \neq \emptyset \), there exist a set of vectors \( \{Y^a, a \in \mathcal{N}(k_0) \cap \mathcal{U} \} \) and a scalar \( \lambda > 0 \) which is independent of \( Y_i, \forall i \in \mathcal{C} \), such that

(i) \( Y^a \) is a vector solution to system \( \mathcal{Y}(\mathcal{K}(a, \mathcal{U}) \setminus \{k_0\}) \), and

(ii) \( Y_{k_0} = \lambda \sum_{a \in \mathcal{N}(k_0) \cap \mathcal{U}} M_{ka} (Y^a_a)^{\mu_{k_0}/\mu_a} \),

where \( Y^a_a \) is the entry in \( Y^a \) corresponding to node \( a, \forall a \in \mathcal{N}(k_0) \cap \mathcal{U} \).

Proof. From (D.1), for given a node \( k \in \mathcal{K}(k_0, \mathcal{U}) \setminus \mathcal{C} \) and \( \mathcal{N}(k) \cap \mathcal{U} \neq \emptyset \) we can write the recursive equation as

\[
Y_k = \sum_{a \in \mathcal{N}(k) \cap \mathcal{U} \atop \pi(a, k_0, \mathcal{U})=0} M_{ka} Y^a_a^{\mu_{k_0}/\mu_a} + \sum_{a \in \mathcal{N}(k) \cap \mathcal{U} \atop \pi(a, k_0, \mathcal{U})=1} M_{ka} Y^a_a^{\mu_{k_0}/\mu_a}. \tag{D.2}
\]

We have \( \pi(k_0, k, \mathcal{U}) = 1 \), so for each \( a \in \mathcal{N}(k) \) if \( \pi(a, k_0, \mathcal{U}) = 1 \) then \( \pi(k, a, \mathcal{U}) = \pi(a, k, \mathcal{U}) = 1 \). According to Lemma D.5 we have \( \mu_a = \mu_k \) and (D.2) can be written as

\[
Y_k = \sum_{a \in \mathcal{N}(k) \cap \mathcal{U} \atop \pi(a, k_0, \mathcal{U})=0} M_{ka} Y^a_a^{\mu_{k_0}/\mu_a} + \sum_{a \in \mathcal{N}(k) \cap \mathcal{U} \atop \pi(a, k_0, \mathcal{U})=1} M_{ka} Y^a_a. \tag{D.3}
\]

In a similar way for each \( Y_a \) such that \( \pi(a, k_0, \mathcal{U}) = 1 \), we can write \( Y_a \) as a sum of two parts as in (D.2). The first part involves nodes that are not connected to \( k_0 \) and the second involves nodes that are connected to \( k_0 \). Hence, the value of \( Y_k \) can be written as

\[
Y_k = \sum_{\sigma \in \Theta(k, a, \mathcal{U}) \atop \pi(a, k_0, \mathcal{U})=0} v(\sigma) Y^a_a^{\mu_{k_0}/\mu_a} + \sum_{\sigma \in \Theta(k, k_0, \mathcal{U}) \atop \pi(a, k_0, \mathcal{U})=0} v(\sigma) Y_{k_0} \tag{D.4}
\]

where \( v(\sigma) = \prod_{i=0}^{J-1} M_{h_i, h_{i+1}} \) given a path \( \sigma = \{h_0, \ldots, h_J\} \).
Now using (D.1) and then (D.3) and (D.4), we can write $Y_{k_0}$ as

$$Y_{k_0} = \sum_{a \in N(k_0) \cap U} M_{k_0a} \pi_{k_0/a} \mu_{k_0/a}$$

$$= \sum_{\pi(a,k_0,U)=0} M_{k_0a} \pi_{k_0/a} \mu_{k_0/a} + \sum_{\pi(a,k_0,U)=1} M_{k_0a} \left( \sum_{\sigma \in \Theta(a,c,U)} v(\sigma) Y_{c}^{\mu_{c}/\mu_c} + \sum_{\sigma \in \Theta(a,k_0,U)} v(\sigma) Y_{k_0} \right)$$

$$= \sum_{\pi(a,k_0,U)=0} M_{k_0a} \pi_{k_0/a} \mu_{k_0/a} + \sum_{\pi(a,k_0,U)=1} M_{k_0a} \sum_{\sigma \in \Theta(a,c,U)} v(\sigma) Y_{c}^{\mu_{c}/\mu_c} + \sum_{\sigma \in \Theta(a,k_0,U)} v(\sigma) Y_{k_0}.$$ 

This is equivalent to

$$\left( 1 - \sum_{\sigma \in \Theta(k_0,k_0,U)} v(\sigma) \right) Y_{k_0} = \sum_{\pi(a,k_0,U)=0} M_{k_0a} \pi_{k_0/a} \mu_{k_0/a} + \sum_{\pi(a,k_0,U)=1} M_{k_0a} \sum_{\sigma \in \Theta(a,c,U)} v(\sigma) Y_{c}^{\mu_{c}/\mu_c} + \sum_{\sigma \in \Theta(a,k_0,U)} v(\sigma) Y_{k_0}.$$ 

If we denote $\lambda = \frac{1}{\sum_{\sigma \in \Theta(k_0,k_0,U)} v(\sigma)}$, then we get

$$Y_{k_0} = \lambda \left( \sum_{\pi(a,k_0,U)=0} M_{k_0a} \pi_{k_0/a} \mu_{k_0/a} + \sum_{\pi(a,k_0,U)=1} M_{k_0a} \sum_{\sigma \in \Theta(a,c,U)} v(\sigma) Y_{c}^{\mu_{c}/\mu_c} \right). \quad (D.5)$$

Indeed, according to (D.5) and the definition of $\lambda$, we have $\lambda > 0$ and it is independent of $Y$, $\forall i \in C$. The lemma can be proved by considering each part in the sum in the parentheses in (D.5) as follows.

We consider the first part of the sum in the parentheses in (D.5). Given a node $a_1 \in N(k_0) \cap U$ with $\pi(a_1,k_0,U) = 0$, we have $k_0 \notin K(a_1,U)$ and consequently $K(a_1,U) \setminus \{k_0\} \equiv K(a_1,U)$. Now we will prove that $(Y_{a_1}, a_1' \in K(a_1,U))$ is a vector solution to the system $Y(K(a_1,U) \setminus \{k_0\})$. Under the hypotheses of the lemma and the definition in (D.1) we have that $(Y_{k'}, k' \in U)$ is a vector solution to $Y(U)$. So given a node $a_1' \in K(a_1,U) \subset U$ we have the following system of equations

$$Y_{a_1'} = \begin{cases} \sum_{c \in N(a')} \pi_{a_1'/c} M_{a_1'/c} \pi_{a_1'/c} & \text{if } a_1' \in U \setminus C \\ e^U \pi_{a_1'} & \text{if } a_1' \in U \cap C \\ 0 & \text{if } a_1' \notin C, N(a') \cap U = \emptyset \end{cases}. \quad (D.6)$$

The following remarks are obviously verified: (i) $N(a') \cap U \equiv N(a') \cap K(a_1,U)$, (ii) $(U \setminus C) \cap
\( \mathcal{K}(a_1, U) \equiv \mathcal{K}(a_1, U) \setminus C \), (iii) \((U \cap C) \cap \mathcal{K}(a_1, U) \equiv \mathcal{K}(a_1, U) \cap C \). From (i), (ii) and (iii), (D.6) can be written as

\[
Y_{a'} = \begin{cases} 
\sum_{c \in \mathcal{N}(a') \cap \mathcal{K}(a_1, U)} M_{a'c} Y_{\bar{c}}^{\mu_{a'}/\mu_c} & \text{if } a' \in \mathcal{K}(a_1, U) \setminus C \\
0 & \text{if } a' \in \mathcal{K}(a_1, U) \cap C \\
\sum_{c \in \mathcal{N}(a') \cap \mathcal{K}(a_1, U)} M_{a'c} Y_{\bar{c}}^{\mu_{a'}/\mu_c} & \text{if } a' \notin C, \mathcal{N}(a') \cap \mathcal{K}(a_1, U) = \emptyset,
\end{cases}
\]

leading to the fact that \((Y_{a'}, a' \in \mathcal{K}(a_1, U))\) is a vector solution to the system \( \mathcal{Y}(\mathcal{K}(a_1, U)) \).

Moreover, as mentioned earlier, \( \mathcal{K}(a_1, U) \setminus \{k_0\} \equiv \mathcal{K}(a_1, U) \), so we can write the first part of the sum in the parentheses in (D.5) as

\[
\sum_{a_1 \in \mathcal{N}(k_0) \cap U} M_{k_0a_1} Y_{\bar{a}}^{\mu_{k_0}/\mu_{a_1}} = \sum_{a_1 \in \mathcal{N}(k_0) \cap U} M_{k_0a_1}(Y_{a_1}^{\mu_{a_1}})^{\mu_{k_0}/\mu_{a_1}}, \tag{D.7}
\]

where \( Y_{a_1} \) is a vector solution to \( \mathcal{Y}(\mathcal{K}(a_1, U) \setminus \{k_0\}) \).

We now consider the second part of the sum in the parentheses in (D.5). Given \( a_2 \in \mathcal{N}(k_0) \cap U \) such that \( \pi(a_2, k_0, U) = 1 \), we define a vector \((\bar{Y}_c, c \in \mathcal{K}(a_2, U) \setminus \{k_0\})\) with entries

\[
\bar{Y}_c = Y_c - \sum_{\sigma \in \Theta(c, k_0, U)} v(\sigma) Y_{k_0}, \quad \forall c \in \mathcal{K}(a_2, U) \setminus \{k_0\}. \tag{D.8}
\]

Based on (D.4), for each \( c \in \mathcal{K}(a_2, U) \setminus \{k_0\}, c \notin C \) and \( \mathcal{N}(c) \cap U \neq \emptyset \) we have

\[
\tilde{Y}_c = \sum_{\sigma \in \Theta(c, k_0, U) \setminus (c, k_0, U)} v(\sigma) Y_d^{\mu_c/\mu_d}
\]

\[
= \sum_{c' \in \mathcal{N}(c) \cap U} M_{cc'} Y_{\bar{c}}^{\mu_c/\mu_{c'}} + \sum_{c' \in \mathcal{N}(c) \cap U} \left( M_{cc'} \sum_{\sigma \in \Theta(c', d, k_0, U) \setminus (c', k_0, U)} v(\sigma) Y_d^{\mu_{c'}/\mu_d} \right) \tag{D.9}
\]

Moreover, from the definition in (D.8) we note that \( \tilde{Y}_c = Y_c \) if \( \pi(c', k_0, U) = 0 \), \( \forall c' \in \mathcal{K}(a_2, U) \), so (D.9) can be written as

\[
\tilde{Y}_c = \sum_{c' \in \mathcal{N}(c) \cap U} M_{cc'} (Y_{c'})^{\mu_c/\mu_{c'}}, \quad \forall c \in \mathcal{K}(a_2, U) \setminus \{C \cup \{k_0\}\}, \mathcal{N}(c) \cap U \neq \emptyset. \tag{D.10}
\]
Moreover, it is obvious to verify that (iv) \( (N(c) \cap U) \setminus \{k_0\} = N(c) \cap K(a_2, U) \setminus \{k_0\} \), (v) if \( c \in C \) then \( \bar{Y}_c = Y_c = e^{Uc} \) and (vi) if \( N(c) \cap U \setminus \{k_0\} = \emptyset \) then \( \bar{Y}_c = 0 \). From (iv), (v), (vi), (D.10) and the definition in (D.1) we have that \( \bar{Y} \) is a vector solution to \( \mathcal{Y}(K(a_2, U) \setminus \{k_0\}) \). In summary, according to (D.9), for each \( a_2 \in N(k_0) \cap U \) and \( \pi(a_2, k_0, U) = 1 \), there exists a vector \( \bar{Y} \) which is a solution to \( \mathcal{Y}(K(a_2, U) \setminus \{k_0\}) \) such that

\[
\bar{Y}_{a_2} = \sum_{\sigma \in \Theta(a_2, c, U), \ c \in K(a_2, U), \ c(k_0, U) = 0} v(\sigma) e^{\mu a_2 / \mu_c}.
\]

Consequently, there exits vectors \( Y^{a_2} \) which are solutions to \( \mathcal{Y}(K(a_2, U) \setminus \{k_0\}) \), \( \forall a_2 \in N(k_0) \cap U, \pi(a_2, k_0, U) = 1 \), such that the second part of the sum in the parentheses in (D.5) can be written as

\[
\sum_{a_2 \in N(k_0, U)} M_{k_0a_2} \sum_{\sigma \in \Theta(a_2, c, U), \ c \in K(a_2, U), \ c(k_0, U) = 0} v(\sigma) e^{\mu a_2 / \mu_c} = \sum_{a_2 \in N(k_0, U)} M_{k_0a_2} Y^{a_2}_{a_2}. \tag{D.11}
\]

Finally, the lemma is proved using (D.5) and (D.7) and (D.11).

\[\square\]

**Lemma D.7.** Under the hypotheses in Theorem 8.2, \( Y_r \) is a \( \mu_r \)--MEV CPGF.

**Proof.** We prove this lemma by using the results from Lemma D.6. We first create a tree structure \( T \) where each node in \( T \) represents a pair \((k, U)\), where \( k \in \mathcal{N} \) and \( U \subseteq \mathcal{N} \). We create nodes in \( T \) recursively as follows.

The root of \( T \) corresponds to \((r, \mathcal{N})\). For each created node \( t(k, U) \in T \), if \( k \in C \) or \( N(k) \cap U = \emptyset \) then we consider \( t(k, U) \in T \) as a leaf of \( T \), i.e., it has no successor. Otherwise, for each node \( a \in N(k) \cap U \) we create a new successor node of \( t \) and associate a pair \((a, K(a, U) \setminus \{k\})\) to the new node. We note that \( N(k) \cap U = \emptyset \) occurs only when \(|U| = 1 \). Figure D.1 shows an example where the root \( r \) has two successors \( a_1, a_2 \) \((a_1, a_2 \in N(r))\).

For each two nodes \( p(k, U) \) and \( q(k', U') \) in \( T \) such that \( q \) is a successor node of \( p \), by definition we have the fact that \(|U| > |U'|\), so \( T \) does not contain any cycle. Moreover, since we stop creating nodes when \(|U| = 1 \), the number of nodes in \( T \) is finite. We denote \( T(p) \) be the set of successor nodes of \( p \) in \( T \). We recursively associate each note \( t(k, U) \in T \) with a value \( Z_t \) such that \( Z_t = Y^k_k \), where \( Y^k \) is a vector solution to \( \mathcal{Y}(U) \), as follows. The root \( r_t \) of \( T \) is associated with \( Y_r \), i.e., \( Z_{r_t} = Y_r \). Given a node \( p(k, U) \in T \) with the corresponding value \( Z_p \). According to Lemma D.6, there exist vectors \( Y^a \), \( \forall a \in N(k) \cap U \), such that

\[
Z_p = \lambda \sum_{a \in N(k) \cap U} M_{ka}(Y^a_a)^{\mu_k / \mu_a}, \tag{D.12}
\]
where $Y^a$ is a vector solution to system $\mathcal{Y}(\mathcal{K}(a, U) \setminus \{k\})$. For each successor node $q(a, \mathcal{K}(a, U) \setminus \{k\})$ of node $p$ we set $Z_q = Y^a_q$ and associate arc $(p, q)$ a non-negative scalar $\eta_{pq} = \lambda M_{ka}$. Note that if $p(k, U)$ is a leaf, i.e. $k \in C$ or $|U| = 1$, then $Z_p = e^{U_k}$ (if $k \in C$) or $Z_p = 0$ (if $k \notin C$ and $|U| = 1$). Moreover, we also associate each node $p(k, U)$ with a positive scalar $\xi_p = \mu_k$. Hence, each node $p \in \mathcal{T}$ is associated with a positive value $Z_p$ and a scalar $\xi_p > 0$. Based on (D.12) the value of $Z_p$ for each node $p(k, U)$ can be defined recursively as

$$Z_p = \begin{cases} 
\sum_{q \in T(p)} \eta_{pq}(Z_q)^{\xi_p/\xi_q} & \text{if } p \text{ is not a leaf} \\
e^{U_k} & \text{if } k \in C \\
0 & \text{if } |U| = 1 \text{ and } k \notin C.
\end{cases}$$  \hspace{1cm} \text{(D.13)}$$

By simply removing nodes $p$ with zero values ($Z_p = 0$) from tree $\mathcal{T}$, (D.13) can be written as

$$Z_p = \begin{cases} 
\sum_{q \in T(p)} \eta_{pq}(Z_q)^{\xi_p/\xi_q} & \text{if } p \text{ is not a leaf} \\
e^{U_k} & \text{otherwise}.
\end{cases}$$  \hspace{1cm} \text{(D.14)}$$

(D.14) is equivalent to the definition of the CPGFs given by (8.1). Accordingly, $Z_r$ is a $\xi_r$-MEV CPGF, or equivalently $Y_r$ is a $\mu_r$-MEV CPGF.

**Corollary D.8.** Under the hypotheses in Theorem 8.2, $Y_k$, $\forall k \in \mathcal{N} \setminus C$, is a $\mu_k$-MEV CPGF function.

**Proof.** It makes no difference for the statement that the graph may contain cycles since we can consider any node $k \in \mathcal{N} \setminus C$ as a root and apply the proof in Lemma D.7 to the sub-graph with nodes $\mathcal{K}(k, \mathcal{N})$ to prove that $Y_k$ is a $\mu_k$-MEV CPGF.

**Lemma D.9.** If $G(y) = Y_r$ is a $\mu_r$-MEV CPGF then the choice probability given by the generating function $G(y_i, i \in C) = Y_r$ is

$$P(i) = \sum_{[k_t, ..., k_0 = i] \in \Phi(i)} \prod_{i=0}^{t-1} P(k_i | k_{i+1}), \quad \forall i \in C.$$
Proof. We derive the probabilities given by an MEV model with the CPGF \( G(y) = G(y_i, i \in C) = Y_r \). Each alternative \( i \) is associated with the utility \( U_i + \epsilon_i \), where vector \( \epsilon \) is MEV distributed with the CPGF \( G(y) \). Since \( Y_r \) is a \( \mu_r - MEV \) CPGF function, the choice probability is (McFadden, 1978)

\[
P(i) = \frac{y_i \frac{\partial G(y)}{\partial y_i}}{\mu_r G(y)} = \frac{y_i}{\mu_r Y_r} \frac{\partial Y_r}{\partial y_i}, \quad i \in C.
\]

From (8.11), the partial derivative of \( Y_k \) with respect to \( y_i, i \in C \) is

\[
\frac{\partial Y_k}{\partial y_i} = \sum_{a \in N(k)} e^{\mu_k v(a|k) Y_a^\mu_k/\mu_a} \frac{\partial Y_a}{\partial y_i} Y_a Y_k, \quad k \in N \setminus C. \tag{D.15}
\]

Denote \( S^i_k = \frac{y_i}{\mu_k Y_k} \frac{\partial Y_k}{\partial y_i}, \quad i \in C, k \in N \). Based on (D.15), we obtain recursive formulas for \( S^i_k \) as

\[
S^i_k = \sum_{a \in N(k)} e^{\mu_k v(a|k) Y_a^\mu_k/\mu_a} \frac{\partial Y_a}{\partial y_i} S^a_i = \sum_{a \in N(k)} P(a|k) S^a_i, \quad \forall k \in N \setminus C.
\]

Note that

\[
S^i_i = \frac{y_i}{\mu_i Y_i} \frac{\partial Y_i}{\partial y_i} = \frac{y_i}{\mu_i Y_i} \mu_i Y_i^{-1} = 1,
\]

and

\[
S^i_j = \frac{y_i}{\mu_i Y_j} \frac{\partial Y_j}{\partial y_i} = 0, \forall j \in C, j \neq i,
\]

so the choice probability given by the MEV model is

\[
P(i) = S^i_r = \sum_{[k_1, \ldots, k_t = i] \in \Phi(i)} \prod_{i=0}^{t-1} P(k_i|k_{i+1}), \quad \forall i \in C.
\]

Since the choice probability given by Lemma D.9 is identical to (8.6), Theorem 8.2 is completely proved.

D.2 Proof of Theorem 8.3

We provide a detailed proof for Theorem 8.3 in this appendix. We first introduce a lemma. For a node \( k \in N \) we define \( l(k) \) be the number of arcs of the longest path (defined by the number of links) from \( k \) to the destinations. We also call \( l(k) \) the level of node \( k \). We then denote \( L(t) \) be the set of nodes of level \( t \): \( L(t) = \{k \in N | l(k) = t\} \). Since the graph is cycle-free, the level of
each node is finite. Moreover, the root has the highest level in the graph. We have the following lemma

**Lemma D.10.** Given $t > 0$, $t \in \mathbb{Z}$, we have $Y_k^{(i)} = Y_k^{(t+1)}$, $\forall i \geq t + 1$ and $\forall k \in \bigcup_{j=1}^{t} \mathcal{L}(j)$.

**Proof.** We prove this lemma by induction. For $t = 1$, $\mathcal{L}(t)$ is the set of nodes that connect directly to the destination. So $\forall i \geq 1$ and $\forall k \in \mathcal{L}(1)$, according to (8.21) we have

$$Y_k^{(i)} = \sum_{a \in \mathcal{C}} M_{ka}(Y_a^{(i)})^{\mu_k/\mu_a}. \quad (D.16)$$

Moreover, from (8.20) and (8.21) we have $Y_a^{(i)} = e^{U_a} \forall a \in \mathcal{C}, i \geq 1$. Substituting into (D.16), we obtain

$$Y_k^{(i)} = \sum_{a \in \mathcal{C}} M_{ka}(e^{U_a})^{\mu_k/\mu_a} = Y_k^{(2)}, \quad \forall i \geq 1, k \in \mathcal{L}(1),$$

meaning that the result is true for $t = 1$. Now we assume that the result is true for $t \geq 1$

$$Y_k^{(i)} = Y_k^{(t+1)}, \forall i \geq t + 1, k \in \bigcup_{j=1}^{t} \mathcal{L}(j), \quad (D.17)$$

leading to the fact that

$$Y_k^{(i)} = Y_k^{(t+2)}, \forall i \geq t + 2, k \in \bigcup_{j=1}^{t} \mathcal{L}(j). \quad (D.18)$$

Given a node $k \in \mathcal{L}(t + 1)$, we note that for each node $a \in \mathcal{N}(k), a \in \bigcup_{j=1}^{t} \mathcal{L}(j)$ (otherwise the level of $k$ would be greater than $t + 1$). From (8.21) we have

$$Y_k^{(i+1)} = \sum_{a \in \mathcal{N}(k)} \sum_{a \in \bigcup_{j=1}^{t} \mathcal{L}(j)} M_{ka}(Y_a^{(i)})^{\mu_k/\mu_a} + b_k, \quad \forall i \geq t + 1. \quad (D.19)$$

Then according to (D.17), (D.19) can be written as

$$Y_k^{(i+1)} = \sum_{a \in \mathcal{N}(k)} \sum_{a \in \bigcup_{j=1}^{t} \mathcal{L}(j)} M_{ka}(Y_a^{(t+1)})^{\mu_k/\mu_a} + b_k = Y_k^{(t+2)}, \quad \forall i \geq t + 1.$$

This means that $Y_k^{(i)} = Y^{(t+2)}, \forall i \geq t + 2, \forall k \in \mathcal{L}(t + 1)$, and from (D.18) we have

$$Y_k^{(i)} = Y_k^{(t+2)}, \forall i \geq t + 2, \forall k \in \bigcup_{j=1}^{t+1} \mathcal{L}(j).$$

This validates the result for $t + 1$, as required. \qed
Now if we choose $K = l(r) + 1$ ($l(r)$ is the level of the root), based on Lemma D.10 we have the following result

$$Y^{(i)}_k = Y^{(K)}_k, \forall i \geq K, \forall k \in \bigcup_{j=1}^{l(r)} L(j),$$

and note that $\bigcup_{j=1}^{l(r)} L(j) \equiv \mathcal{N}$. Hence, Theorem 8.3 is proved.

### D.3 Derivatives of the Log-likelihood Function

In this appendix, we derive the derivatives of the log-likelihood function defined in (8.26). The gradient of the choice probability $P(i_n|C_n)$ can be obtained by taking the Jacobian of vector $F$ which can be derived based on (8.24). The Jacobian of $F$ with respect to parameter $\beta_j$ is

$$\frac{\partial F}{\partial \beta_j} = (I - P^T)^{-1} \frac{\partial P^T}{\partial \beta_j} F. \quad (D.20)$$

Hence it requires the first derivative of each element of matrix $P$ with respect to parameter $\beta_j$. Note that

$$P_{ka} = P(a|k) = M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k}, \forall k, a \in \mathcal{N}. \quad (8.21)$$

We take the derivative of a given $P_{ka}$ and obtain

$$\frac{\partial P_{ka}}{\partial \beta_j} = \frac{\partial M_{ka} Y_a^{\phi_{ka}}}{\partial \beta_j} \frac{Y_a}{Y_k} - M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k^2} \frac{\partial Y_k}{\partial \beta_j}$$

$$+ M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k} \left( \frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a + \frac{\phi_{ka}}{Y_a} \frac{\partial Y_a}{\partial \beta_j} \right).$$

Hence it requires the derivative of $Y_k$, $\forall k \in \mathcal{N}$. We take the derivative of a given value $Y_k$, $k \in \mathcal{N}$ as defined by (8.21) and obtain

$$\frac{\partial Y_k}{\partial \beta_j} = \sum_{a \in \mathcal{N}} \left( \frac{\partial M_{ka} Y_a^{\phi_{ka}}}{\partial \beta_j} + M_{ka} Y_a^{\phi_{ka}} \left( \frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a + \frac{\phi_{ka}}{Y_a} \frac{\partial Y_a}{\partial \beta_j} \right) \right) + \frac{\partial b_k}{\partial \beta_j}. \quad (D.21)$$

We introduce two matrices $S$ and $H$ of size $|\mathcal{N}| \times |\mathcal{N}|$ with entries

$$S_{ka} = \frac{\partial M_{ka} Y_a^{\phi_{ka}}}{\partial \beta_j} + M_{ka} Y_a^{\phi_{ka}} \frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a \quad \forall k, a \in \mathcal{N},$$

$$H_{ka} = M_{ka} Y_a^{\phi_{ka}} \frac{\phi_{ka}}{Y_a}$$

So (D.21) becomes

$$\frac{\partial Y_k}{\partial \beta_j} = \frac{\partial b_k}{\partial \beta_j} + \sum_{a \in \mathcal{N}(k)} \left( S_{ka} + H_{ka} \frac{\partial Y_a}{\partial \beta_j} \right), \quad \forall k \in \mathcal{N}.$$
This allows us to define the Jacobian of vector $Y$ as a system of linear equations

$$
\frac{\partial Y}{\partial \beta_j} = Se + H \frac{\partial Y}{\partial \beta_j} + \frac{\partial b}{\partial \beta_j} \Rightarrow \frac{\partial Y}{\partial \beta_j} = (I - H)^{-1}(Se + \frac{\partial b}{\partial \beta_j}).
$$

As suggested by Mai et al. (2015c) we can derive the Jacobian of $V$ instead of $Y$ to avoid numerical issues. Note that $Y_k = e^{\mu_k}V(k)$, the gradient of $Y_k$ with respect to $\beta_j$, can be written as

$$
\frac{\partial Y_k}{\partial \beta_j} = \frac{\partial V(k)}{\partial \beta_j} \mu_k Y_k - \frac{\partial \mu_k}{\partial \beta_j} V(k)Y_k \quad \forall k \in \mathcal{N}.
$$

Using (D.21) and (D.23) we obtain

$$
\frac{\partial V(k)}{\partial \beta_j} = \sum_{a \in \mathcal{N}} R_{ka} + \sum_{a \in \mathcal{N}} L_{ka} \frac{\partial V(a)}{\partial \beta_j} + h_k \quad \forall k \in \mathcal{N},
$$

where

$$
\begin{align*}
R_{ka} &= \frac{1}{\mu_k} \frac{\partial M_{ka} Y_a^{\phi_{ka}}}{\partial \beta_j} Y_k + \frac{1}{\mu_k} M_{ka} Y_a^{\phi_{ka}} \frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a - \mu_k M_{ka} V(a) Y_a^{\phi_{ka}} \frac{\partial \mu_a}{\partial \beta_j}, \\
L_{ka} &= M_{ka} Y_a^{\phi_{ka}} Y_k, \\
h_k &= \frac{1}{\mu_k Y_k} \frac{\partial b_k}{\partial \beta_j} + \mu_k V(k) \frac{\partial \mu_k}{\partial \beta_j}.
\end{align*}
$$

Let $R, L$ be two matrices and $h$ be a vector of size $|\mathcal{N}| \times |\mathcal{N}|$, $|\mathcal{N}| \times |\mathcal{N}|$, $|\mathcal{N}|$, with entries $R_{ka}, L_{ka}$ and $h_k$, $\forall k,a \in \mathcal{N}$, respectively. The Jacobian of the vector of value functions can be written as a linear system

$$
\frac{\partial V}{\partial \beta_j} = (I - L)^{-1}(Re + h).
$$

Although (D.24) and (D.22) are equivalent, there are numerical considerations. On the one hand, each element of matrix $L$ is defined as $L_{ka} = M_{ka} Y_a^{\phi_{ka}} Y_k$ and according to (8.21) we have $Y_k > M_{ka} Y_a^{\phi_{ka}} > 0$, leading to the fact that the elements of $L$ vary in $(0, 1)$. On the other hand, each element of $H$ is $H_{ka} = M_{ka} Y_a^{\phi_{ka} - 1} \phi_{ka}$ and varies in $(0, \infty)$. So the elements of matrix $L$ are closer in value, compared to matrix $H$, meaning that using (D.24) to compute the gradient of LL function is better than (D.22) for numerical reasons. Note that Mai et al. (2015c) has a similar conclusion when comparing two formulas of the derivative of the value functions in route choice applications.

We note that the derivative of each element of matrix $M$ with respect to parameter $\beta_j$ is

$$
\frac{\partial M_{ka}}{\partial \beta_j} = M_{ka} \left( \frac{\mu_k \partial v(a|k)}{\partial \beta_j} + v(a|k) \frac{\partial \mu_k}{\partial \beta_j} \right), \quad \forall k, a \in \mathcal{N}.
$$

In summary, the derivatives of the model have complicated form but can be quickly computed for large-scale problems using the linear systems in (D.20) and (D.24). The model derivatives allow us to use classic Hessian approximations such as BHHH and BFGS (see for instance Berndt
et al., 1974, Nocedal and Wright, 2006) to maximize the log-likelihood function. We note that Eberwein and Ham (2008b) also derive the analytical derivatives for a general dynamic discrete choice model. These formulas are however different with the ones derived in this section due to the different definitions of the log-likelihood functions.

### D.4 Elasticities

The elasticity of demand for alternative $i$ with respect to an attribute $x_j$ of alternative $j$ is

$$e_{i,x_j} = \frac{\partial P(i)}{\partial x_j} \frac{x_j}{P(i)} = \frac{\partial P(i)}{\partial x_j} \frac{x_j}{P(i)}, \quad i, j \in C.$$ 

If the utility $U_j$ is linear in $x$, $\frac{\partial U_j}{\partial x_j}$ is a constant. We now analyze the model structure in terms of the sensitivity of demand to changes in the utility of alternatives $\frac{\partial P(i)}{\partial U_j}$. Similarly to the previous section we derive formulas for the elasticity of demand so that they can be computed quickly.

Note that $\frac{\partial P(i)}{\partial U_j} = \frac{\partial F}{\partial U_j}$ and the Jacobian of vector $F$ with respect to $U_j$ can be derived using (8.24)

$$\frac{\partial F}{\partial U_j} = (I - P^T)^{-1} \frac{\partial P^T}{\partial U_j} F,$$  \hspace{1cm} (D.25)

Using (8.12), the derivative of an element $P_{ka}$, $k, a \in \mathcal{N}$ with respect to $U_j$ is

$$\frac{\partial P_{ka}}{\partial U_j} = P_{ka} \left( \phi_{ka} \frac{\partial Y_a}{\partial U_j} - \frac{\partial Y_k}{Y_k \partial U_j} \right),$$

where $\phi_{ka} = \mu_k / \mu_a$. Hence it requires the first derivatives of $Y_k$, $\forall k \in \mathcal{N}$, with respect to $U_j$. Taking the derivative of (8.21) with respect to $U_j$, we obtain

$$\frac{\partial Y_k}{\partial U_j} = \sum_{a \in \mathcal{N}} \phi_{ka} M_{ka} Y_a^{\phi_{ka}-1} \frac{\partial Y_a}{\partial U_j} + \frac{\partial b_k}{\partial U_j}, \quad \forall k \in \mathcal{N}.$$ 

And we note that

$$\frac{\partial b_k}{\partial U_j} = \begin{cases} 0 & \text{if } k \neq j \\ Y_j / \mu_j & \text{if } k = j \end{cases}.$$ 

So if we denote a matrix $T(|\mathcal{N}| \times |\mathcal{N}|)$ with entries $T_{ka} = \phi_{ka} M_{ka} Y_a^{\phi_{ka}-1}$, $\forall k, a \in \mathcal{N}$, then the Jacobian of vector $Y$ can be written as system of linear equations

$$\frac{\partial Y}{\partial U_j} = (I - T)^{-1} d,$$  \hspace{1cm} (D.26)
where $d$ is a vector of size $|\mathcal{N}|$ with zero values for all nodes except for node $j$ that equals $Y_j/\mu_j$. Therefore, the elasticity of demand for alternative $i$ with respect to an attribute $x_j$ can be computed by solving the linear systems (D.25) and (D.26).
Bibliography


