UCGE Reports
Number 20349

Department of Geomatics Engineering

Local alignment applied to Grizzly bear GPS tracking data.

by

Andrés R. Ortíz Villagómez.

April, 2012

UNIVERSITY OF CALGARY

Local alignment applied to Grizzly bear GPS tracking data.

by

Andrés R. Ortíz Villagómez

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

April, 2012

Abstract

Recent advances in GPS collar technologies for Grizzly bear tracking have produced a drastic increase in the volume of data available for scientific analysis. Machine learning methods seem suited to process this ever-increasing volume of data. Comprehensive understanding of the datasets, machine learning methods and similarity measures is fundamental for research of this kind.

To automatically detect frequent movement patterns, the current work implemented three machine learning methods, a Location-Based Services (LBS), a simulated annealing, and a hybrid local alignment approach. Several dataset segmentations were tested to reduce the amount of calculations for similarity measures, without losing relevant data relationships.

Mostly based on my fifteen years of professional experience in the industry of database administration and development, I found the current state of commercial database management systems (DBMS) mature enough to conduct fully integrated implementations. In my judgment, that assumption was validated by the results of the local alignment method.

Acknowledgements

I would like to thank the following individuals for their contributions to my work.

First and foremost, to my supervisor, Dr. Andrew J. S. Hunter, for giving me the opportunity to work on this project. His contributions to the very early stages of this research have been of remarkable value.

The Foothills Research Institute for providing the grizzly bear tracking dataset used in this work.

To Dr. Xin Wang, for her guidance on finding some of the most recent and best quality contributions to the Spatial Databases and Data Mining research literature.

For reminding me with his excellent lectures that, precision and accuracy are still very much at the heart of exact sciences and engineering, I owe Dr. Derek Lichti a debt of gratitude. His encouragement for diligent methodical work at a critical point in this process, allowed me to continue when I felt overwhelmed.

To Dr. Danielle Marceau, her inspirational lectures on new ways of reasoning about the use of computers and computational processes in science, has opened a vast new horizon for me. The ideas emanating from those concepts seem inexhaustible.

For his comprehensive technical lectures on the use of remotely sensed data for geographic and environmental research, Dr. Quazi Hassan has made me think about digital cartography in a very different light.

To all other faculty members of the Department of Geomatics Engineering who allowed me to audit their lectures. I believe that what I have learned in your classrooms has been put to good use.

Many of my fellow graduate students in the department allowed me to ask them innumerable questions, technical and otherwise. Their opinions represented a solid sounding board for many of my ideas in this specialty.

The care and unconditional support of my sister Beatriz, my magnetic Aida, my friend Sandra, my aunt Silvia, my uncle Edmundo, and my brother Armando in Mexico City, my dearest friend Laura in Montreal, my family in Texas, Tom and Mo, my niece Linn in Norway, my brother Gerardo in Scotland, and my cousins Lariza and Eduardo in Toronto, deserve my deepest gratitude.

The patience and encouragement of many other members of my family, genetic and extended alike, past mentors, previous supervisors, and friends, have had a positive impact in these few years of renewed academic work.

The present volume could not have been produced without your contributions. Thank you.

Finally, let me clarify that, any and all mistakes and shortcomings in this volume, are exclusively my responsibility.

Andrés R. Ortíz Villagómez.

Calgary, Canada.

December, 2011.

Dedication

With all my love, to the memory of my mother,

Maria Margarita Villagómez Estrada.

Hay vamos Eñoda ... hay vamos.

Así me hizo mi 'amá, grandote y a lo bruto.

Andrés.

Table of Contents

Chapter **Page**

Abstract **iii**

Acknowledgements **v**

Dedication **vii**

Table of Contents **ix**

List of Tables **xiii**

List of Figures **xvii**

List of Algorithms **xix**

List of Acronyms **xxi**

1 Introduction **1**
- 1.1 Problem definition. 2
- 1.2 Significance. 3
- 1.3 Objectives of this research. 4
- 1.4 Research questions. 5
- 1.5 Research methodology. 6
- 1.6 Scope and limitations. 7
- 1.7 Thesis organization. 8
- 1.8 Chapter summary. 10

2 Literature review **11**
- 2.1 Grizzly bear tracking, habitat, and behavior. 13
  - 2.1.1 Tracking and habitat mapping studies. 13
  - 2.1.2 Habitat, movement and activities. 14
  - 2.1.3 Tracking with GPS instruments. 15
  - 2.1.4 Mathematical models. 17
  - 2.1.5 Conservation. 19
- 2.2 Overview of the evolution of databases and database servers. 19
- 2.3 Knowledge discovery and data mining. 24
- 2.4 Pattern identification with data mining methods. 26
  - 2.4.1 Sequence alignment. 29
    - 2.4.1.1 Local alignment. 30
    - 2.4.1.2 Local alignment in Biotechnology. 34
    - 2.4.1.3 Keyword trees. 39
- 2.5 Analysis theories of moving objects. 52

2.6 Chapter summary. 52

3 Methods 57

3.1 The overall approach. 57
3.2 Study area and tracking dataset. 58
3.2.1 Physical description. 59
3.3 Data cleaning. 62
3.3.1 Conceptual temporal resolution of the cleaned dataset. 65
3.4 Data integration. 66
3.5 Data selection. 68
3.6 Data transformation. 72
3.6.1 Segmentations. 72
3.6.1.1 Annual and seasonal. 73
3.6.1.2 Monthly. 73
3.6.1.3 Weekly. 73
3.6.1.4 Circadian and diel. 73
3.7 Data mining. 78
3.7.1 Local alignment. 78
3.7.1.1 The initial conceptual diagram. 79
3.7.2 The proposed new algorithm. 82
3.7.2.1 Additional improvements. 84
3.7.2.2 Design subsequences. 84
3.7.2.3 Reversed keyword trees for the matching algorithm. 85
3.7.2.4 Mismatch matrix. 91
3.7.2.5 Forwardtrack. 94
3.7.2.6 Matching algorithm. 94
3.7.2.7 Computational complexity. 96
3.8 Pattern evaluation. 98
3.9 Knowledge presentation. 99
3.10 Chapter summary. 100

4 Results and analysis 103

4.1 Benchmark platform. 103
4.2 Local alignment. 103
4.2.1 Memory and secondary storage space requirements. 105
4.2.2 Performance. 106
4.2.3 Performance improvement. 111
4.2.4 Most frequent patterns. 115
4.2.5 Behavioral interpretation of some frequent patterns. 122
4.2.6 Graphic representation of some frequent patterns. 122
4.2.7 Comparison with other methods. 124
4.3 Chapter summary. 125

5 Conclusions **127**
5.1 Answers to the research questions. 128
5.2 Contributions. 130
5.3 Future work. 134
5.3.1 Reorganized reversed keyword trees (RRKTs). 136
5.3.1.1 RRKTs for the design subsequences. 137
5.3.1.2 Failure links for RRKTs. 141

Bibliography **145**

Appendix **Page**

A Initial analysis methods. **177**
A.1 Location-Based Services (LBS). 177
A.1.1 Implementation. 180
A.1.2 Results. 183
A.2 Simulated annealing. 184
A.2.1 Implementation. 186
A.2.2 Results. 189

B Generalizing moving objects and patterns. **191**
B.1 Topological perspective. 192
B.2 Combining topology and behavior. 196
B.3 Trajectory and pattern perspective. 200
B.4 Other perspectives. 209

C Calibration dataset and tables of Grizzly bear tracking data. **211**
C.1 Calibration dataset. 211
C.2 χ^2 test. 213
C.3 Simple segmentations. 213
C.4 Temporal distribution of GPS data. 215

D Most frequent patterns, frequency tables for nine sub variations. **233**
D.1 Original symmetric. 233
D.2 Distinct asymmetric. 235
D.3 Distinct symmetric. 236
D.4 Artificial, repeated asymmetric. 238
D.5 Artificial, repeated symmetric. 239
D.6 Artificial, distinct asymmetric. 241
D.7 Artificial, distinct symmetric. 242
D.8 Completed asymmetric. 244
D.9 Completed symmetric. 245

List of Tables

2.1 Keyword trees' terminal symbols (alphabet) 40
2.2 Fully populated keyword tree, production rules 43
2.3 Uniformly populated keyword tree, production rules 45
2.4 Sparsely populated keyword tree, production rules 49
2.5 Distinctively populated keyword tree, production rules 49

3.1 Study area, natural regions 62
3.2 Land cover, relative abundance 68
3.3 Circadian segmentation, concatenation examples (a) 78
3.4 Circadian segmentation, concatenation examples (b) 78
3.5 Initial conceptual tree, production rules 82
3.6 Design subsequences, sorted 85
3.7 Reversed design subsequences, sorted 85
3.8 Bad character rule matrix 93

4.1 Original asymmetric tree, full failure links by matching symbols, n = 9 . 106
4.2 Performance, original symbols 108
4.3 Performance, distinct symbols 108
4.4 Performance, artificial repeated symbols 108
4.5 Performance, artificial distinct symbols 109
4.6 Performance, completed symbols 109
4.7 Number of individual elements to process, n = 9 114
4.8 Improved execution times, n = 9 115
4.9 Equivalents, land cover symbols to land cover classes 116
4.10 Orig_Asy, most frequent patterns, n = {3, 6, 9, 12} 118
4.11 Orig_Asy, most frequent patterns, n = {15, 20} 118
4.12 Orig_Asy, most frequent patterns, n = {25, 30} 118
4.13 Orig_Asy, most frequent patterns, n = 40 119
4.14 Individual symbol frequencies by segmentation type 120
4.15 Best performance, n = 9 125

5.1 Override comparison vectors, sorted 141

B.1 STN terminal symbols 198
B.2 STN production rules 198
B.3 STN L_D-region topological interactions 199

C.1 Calibration dataset 212
C.2 χ^2 test 213
C.3 Bear population, elevation and geometric area 214
C.4 Bear population by gender 214
C.5 Bear population by elevation 214

C.6 Bear population by gender and elevation 215
C.7 Weekly GPS point distribution, 1999-2003 (a) 216
C.8 Weekly GPS point distribution, 1999-2003 (b) 217
C.9 Weekly GPS point distribution, 2004-2008 (a) 218
C.10 Weekly GPS point distribution, 2004-2008 (b) 219
C.11 Monthly GPS point distribution, 1999-2003 220
C.12 Monthly GPS point distribution, 2004-2008 221
C.13 Individual bear GPS point distribution, 1999 222
C.14 Individual bear GPS point distribution, 2000 223
C.15 Individual bear GPS point distribution, 2001 224
C.16 Individual bear GPS point distribution, 2002 225
C.17 Individual bear GPS point distribution, 2003 226
C.18 Individual bear GPS point distribution, 2004 227
C.19 Individual bear GPS point distribution, 2005 228
C.20 Individual bear GPS point distribution, 2006 229
C.21 Individual bear GPS point distribution, 2007 230
C.22 Individual bear GPS point distribution, 2008 231

D.1 Orig_Sym, most frequent patterns, n = {3, 6, 9, 12} 233
D.2 Orig_Sym, most frequent patterns, n = {15, 20} 234
D.3 Orig_Sym, most frequent patterns, n = {25, 30} 234
D.4 Orig_Sym, most frequent patterns, n = 40 234
D.5 Dist_Asy, most frequent patterns, n = {3, 6, 9, 12} 235
D.6 Dist_Asy, most frequent patterns, n = {15, 20} 235
D.7 Dist_Asy, most frequent patterns, n = {25, 30} 236
D.8 Dist_Sym, most frequent patterns, n = {3, 6, 9, 12} 236
D.9 Dist_Sym, most frequent patterns, n = {15, 20} 237
D.10 Dist_Sym, most frequent patterns, n = {25, 30} 237
D.11 ArtR_Asy, most frequent patterns, n = {3, 6, 9, 12} 238
D.12 ArtR_Asy, most frequent patterns, n = {15, 20} 238
D.13 ArtR_Asy, most frequent patterns, n = {25, 30} 239
D.14 ArtR_Asy, most frequent patterns, n = 40 239
D.15 ArtR_Sym, most frequent patterns, n = {3, 6, 9, 12} 239
D.16 ArtR_Sym, most frequent patterns, n = {15, 20} 240
D.17 ArtR_Sym, most frequent patterns, n = {25, 30} 240
D.18 ArtR_Sym, most frequent patterns, n = 40 240
D.19 ArtD_Asy, most frequent patterns, n = {3, 6, 9, 12} 241
D.20 ArtD_Asy, most frequent patterns, n = {15, 20} 241
D.21 ArtD_Asy, most frequent patterns, n = {25, 30} 242
D.22 ArtD_Sym, most frequent patterns, n = {3, 6, 9, 12} 242
D.23 ArtD_Sym, most frequent patterns, n = {15, 20} 243
D.24 ArtD_Sym, most frequent patterns, n = {25, 30} 243
D.25 Comp_Asy, most frequent patterns, n = {3, 6, 9, 12} 244
D.26 Comp_Asy, most frequent patterns, n = {15, 20} 244

D.27 Comp_Asy, most frequent patterns, n = {25, 30} 245
D.28 Comp_Asy, most frequent patterns, n = 40 245
D.29 Comp_Sym, most frequent patterns, n = {3, 6, 9, 12} 245
D.30 Comp_Sym, most frequent patterns, n = {15, 20} 246
D.31 Comp_Sym, most frequent patterns, n = {25, 30} 246
D.32 Comp_Sym, most frequent patterns, n = 40 246

List of Figures

2.1 Sequence alignment 30
2.2 Algorithm key concepts, Boyer-Moore and Aho-Corasick 33
2.3 Log-likelihood matrix, 3D symmetry analysis 39
2.4 Fully populated keyword tree 42
2.5 Fully populated keyword tree, right failure links 42
2.6 Fully populated keyword tree, left failure links 43
2.7 Uniformly populated keyword tree 44
2.8 Uniformly populated keyword tree, right failure links 44
2.9 Uniformly populated keyword tree, left failure links 45
2.10 Sparsely populated keyword tree 46
2.11 Sparsely populated keyword tree, single-level right failure links 47
2.12 Sparsely populated keyword tree, single-level left failure links 47
2.13 Sparsely populated keyword tree, multi-level right failure links 48
2.14 Sparsely populated keyword tree, multi-level left failure links 48
2.15 Distinctively populated keyword tree 50
2.16 Distinctively populated keyword tree, right failure links 50
2.17 Distinctively populated keyword tree, left failure links 51

3.1 Geographical location of the province of Alberta, Canada 60
3.2 Study area, main towns, roads, rivers and natural regions 61
3.3 Trajectories, bear G001 63
3.4 Frequency distribution of trajectories' time intervals 66
3.5 Study area, elevation and land cover 67
3.6 GPS points by day of the year 70
3.7 GPS points by month of the year 70
3.8 Circadian segmentation, multi-temporal data 75
3.9 Initial conceptual diagram, construction 80
3.10 Initial conceptual diagram 81
3.11 Key conceptual sources 83
3.12 Keyword and reversed-keyword tree, design subsequences 86
3.13 Reversed keyword tree for design subsequences 87
3.14 Reversed keyword tree for design subsequences, Rids and concatenations 88
3.15 Left-pointing failure links for four example nodes 89
3.16 Best equivalent full failure links for four example nodes 92
3.17 Reversed keyword tree for design subsequences, best full failure links . . 92

4.1 Full failure links by tree type and level, n = 9 107
4.2 Equivalent full failure links by tree type and matching symbols, n = 9 . . 107
4.3 Eligible subsequences by keyword length 110
4.4 Keywords searched 110
4.5 Overlapping matches found (a) 112

4.6 Overlapping matches found (b) 112
4.7 Execution times, matching 113
4.8 Most frequent patterns, original asymmetric, n = 3 119
4.9 Most frequent patterns identified for bear G001, n = 6 123

5.1 Reorganized reversed keyword tree by rarest symbol and closer to root . 138
5.2 RRKT by full subsequence probability 138
5.3 RRKT by combined tree level probability 139
5.4 RRKT by tree level, rarest individual symbol 139
5.5 RRKT by tree level, most common symbol 140
5.6 RRKT by tree level, sorted highest to lowest probability 140
5.7 RRKT, first variation, subsequences 142
5.8 RRKT, first variation, full failure links 142

A.1 Location-based services 180

B.1 Generic movement patterns 202
B.2 Leadership pattern between two moving objects 203

List of Algorithms

3.1 Calibration dataset selection 71
3.2 Creation of equivalent full failure links with Forwardtrack, SQL 95
3.3 Loading of equivalent full failure links with Forwardtrack, SQL 95
3.4 Same character detection. Forwardtrack, SQL 95
3.5 *Examine_FLinks* 96
3.6 *Examine_Earlier_FLinks* 96
3.7 *Dictionary_Exact_Matches* 97
A.1 Modified LBS-Alignment similarity measure 182
A.2 Simulated annealing 187

List of Acronyms

ACNg	action-augmented conceptual neighborhood graph. 196
ANSI	American National Standards Institute. 5, 6, 19, 94, 102
ArtD_Asy	Artificial, distinct segmentation, asymmetric log-likelihood ranking.. 104
ArtD_Sym	Artificial, distinct segmentation, symmetric log-likelihood ranking.. 104
ArtR_Asy	Artificial, repeated segmentation, asymmetric log-likelihood ranking.. 104
ArtR_Sym	Artificial, repeated segmentation, symmetric log-likelihood ranking.. 104
BLAST	Basic Local Alignment Search Tool. 36, 37
BLOSUM	Blocks Substitution Matrix. 37
BMI	Behavior Monitoring and Interpretation. 191
Comp_Asy	Completed segmentation, asymmetric log-likelihood ranking.. 104
Comp_Sym	Completed segmentation, symmetric log-likelihood ranking.. 104
CPU	Central Processing Unit. 22
DBA	Database Administrator. 111, 114, 115, 124, 128
DBMS	Database Management System. iii, 12, 22, 57
DBSCAN	Density-Based Spatial Clustering of Applications with Noise. 99, 123
DCB	Dynamic Collective Behavior. 204
DEM	Digital Elevation Model. 66
Dist_Asy	Distinct segmentation, asymmetric log-likelihood ranking.. 104
Dist_Sym	Distinct segmentation, symmetric log-likelihood ranking.. 104
DNA	Deoxyribonucleic Acid. 28, 34, 35
DRA	Dipole Relation Algebra. 196
FASTA	FASTP for all alphabets. 35, 37
FASTP	Protein Sequence Alignment Software. 35
GB	Gigabytes. 23, 57, 103
GHz	Gigahertz. 103
GIS	Geographic Information System. 191, 194
GPS	Global Positioning System. iii, 1, 2, 8, 15–18, 53, 58, 59, 62–69, 72, 74–76, 100, 113, 122, 131, 136, 210, 211, 213, 215
IBE	Interior-Boundary-Exterior. 197, 200

IEC	International Electromechanical Commission. 5, 6, 19, 94, 102
IMB	Individual Movement Behavior. 204, 205
ISO	International Organization for Standardization. 5, 6, 19, 94, 102
KDD	Knowledge Discovery in Databases. 24, 53, 54, 57, 78, 100
LBS	Location-Based Services. iii, 12, 53, 82, 101, 103, 177–179, 208
L_D	Directed Line. 195
MBR	Minimum Binding Rectangle. 99
MCB	Momentary Collective Behavior. 204, 205
MOPS	Millions of Operations Per Second. 125
nDNA	Nuclear Deoxyribonucleic Acid. 18
NP-Complete	Non-deterministic Polynomial Complete. 184
OCV	Override Comparison Vector. 136, 141
$OPRA_m$	Oriented Point Relation Algebra. 196
$OPRA_m^*$	Refined Oriented Point Relation Algebra. 196
Orig_Asy	Original segmentation, asymmetric log-likelihood ranking.. 104
Orig_Sym	Original segmentation, symmetric log-likelihood ranking.. 104
PAM	Mutation Probability Matrix. 36, 37
RAID	Redundant Array of Independent Disks. 22, 23
RAM	Random Access Memory. 2, 5, 22, 23, 96, 103, 105, 129, 131, 132, 134, 186
RDBMS	Relational Database Management System. 4, 6–8, 19, 53, 102, 103, 105, 128, 129, 131, 134, 177, 186
REMO	RElative MOtion. 200–202
Rid	Record identifier. 84, 87
RKT	Reversed Keyword Tree. 58, 87, 93, 96, 98, 104, 105, 131–133
RRKT	Reorganized Reversed Keyword Tree. 136, 137, 141
RSF	Resource Selection Function. 15, 16
RTS	Rule Transition System. 196
SQL	Structured Query Language. 5, 6, 19, 57, 94, 102, 103
SSF	Step Selection Function. 15
STN	Start-Transition-eNd. 197, 198
T-SQL	Transact-SQL. 57, 94
TB	Terabytes. 23
UNIVAC	Universal Automatic Computer. 34

UTM N11	Universal Transverse Mercator N11. 59
UTM N15	Universal Transverse Mercator N15. 59

Chapter 1

Introduction

In order to gain a better understanding of the activities and preferences of wildlife species, scientific studies have made use of the best technologies of their time to track individuals in the studied population. The overall objective of this work is the identification of specific behavior patterns of tracked individuals.

In the past six decades, the tracking methods have improved from the manual efforts of the '50s and '60s, to today's capturing, collaring and release practices. These improvements imposed the need to eliminate many subjective analysis decisions used in the early studies. As a consequence, improving the objectivity of the analysis improved the objectivity of the conclusions of the studies themselves [75]. In practical terms, objective improvement meant an increase in precision, accuracy and frequency of the tracking data available for study, automatically increasing the overall amount of data to examine.

Since human factors involved in data analysis are also questionable in terms of their objectivity, and human reliability decreases significantly in the face of large volumes of repetitive tasks [145], the use of special purpose machines for capture and analysis of tracking datasets became necessary.

As it pertains to the tracking of wild grizzly bears (*Ursus arctos*), the contemporary Global Positioning System (GPS) collar has proved to be more reliable and offers more detailed data, when compared to previous "on-the-ground" and radio tag collar tracking methods of past decades. Given the large volume of data resulting from the new tracking practices, the general purpose programmable computer represents the best alternative currently available for analysis.

This chapter includes descriptions of this research work in terms of its problem definition, significance, objectives, methodology, scope and limitations. The last two sections of this chapter include a description of the rest of this thesis.

1.1 Problem definition.

As will be examined in more detail in section 2.1, the improvements in GPS technology, both in the constellation of satellites, and in the receivers themselves, have allowed scientists to equip grizzly bears with smaller, lighter and more reliable instruments that are capable of acquiring consecutive GPS fixes more frequently over longer periods.

Comparing the collars that were programmed to acquire a satellite fix every hour to the collars that acquire a position every twenty minutes, the volume of captured data has increased three-fold. Translating this increase over the entire tracking season for all bears included in a study, means tens of thousands of individual position fixes per year. As such, efficient analysis of this volume of data to detect repeated movement patterns across the entire population of bears, can only be undertaken with automated computer methods.

The problem then becomes one of creating computer programs that need a minimum amount of input from the user, that analyze a variety of patterns (e.g. statistically rare and common alike), and that are reasonably scalable to be applied to datasets of any size. In other words, the key problem is to create analysis programs that need the bare minimum of user parameters, that simultaneously test for different patterns in the data, and that are reasonably scalable, both, in terms of execution time, and required Random Access Memory (RAM) and secondary storage space.

1.2 Significance.

Compared to the previous generation of computational search algorithms, machine learning and data mining methods expand the search parameters from a single hard-coded pattern (some times several patterns searched in serial manner) to a range of patterns defined with abstract mathematical formulas (e.g. cost functions, geographic proximity, direction of travel, entropy) [128]. Therefore, the effectiveness of these methods depends on the specific similarity measure employed [128]. Comprehensive understanding of the datasets, computational methods, and similarity measures, is fundamental for all research of this kind.

The population of grizzly bears in western Canada is under intense study. The main reason for this level of attention is that ecologists and environmental experts have identified the grizzly bear as a species highly susceptible to modifications in its habitat [73, 112, 114]. Therefore, a measure of how well or poorly the grizzly bear population is doing at any given time gives a strong indication of how the entire ecosystem is doing [129, 165, 176, 193].

Identifying frequent movement patterns for grizzly bears will provide strong indications for provincial administrators and conservation organizations so that they can determine the most effective measures to regulate and protect the ecosystem, and grizzly bears specifically. In terms of governance, it will allow taxpayers's money to be spent on specific protection and conservation initiatives with a reasonable amount of objective justification, and as a consequence, with a reasonable expectation of success.

Once validated, the analysis methods resulting from this type of study can be adapted to other wildlife species tracked in similar ways. Furthermore, once algorithms have been developed that identify the special conditions for grizzly bears in specific geographical regions, modifications to algorithms for other geographic areas and other species should require less effort.

The re-utilization of these methods will help determine the quality and relevance of the original programs in conditions other than their initial intended application. From these "learned lessons", the programs can be deemed useful and improved, or labeled as ineffective and discarded as irrelevant and/or unnecessary.

1.3 Objectives of this research.

1. Find frequent movement patterns in a multi-year grizzly bear tracking dataset.

2. Identify environmental factors that have a larger influence on grizzly bear activities and geographical preference, in order to include them as part of the method's evaluation criteria to obtain a more comprehensive analysis tool.

3. Identify at least one machine learning method suitable for the analysis of grizzly bear tracking data.

4. Implement the entire method using only the internal resources of a Relational Database Management System (RDBMS).

5. Describe the method in simplified terms, preferably with graphical illustrations of its most complex concepts.

6. Find a topological analysis theory capable of explaining all aspects of moving object patterns based on fundamental movement characteristics (e.g. velocity, direction of travel).

Clarifying point number 4 of the previous list, adopt a minimalistic implementation philosophy as defined by the American National Standards Institute (ANSI) / International Organization for Standardization (ISO) / International Electromechanical Commission (IEC) Structured Query Language (SQL) specifications [13, 14, 15, 16, 17, 18, 19, 20, 21, 22].

1.4 Research questions.

1. What are the environmental factors that have a greater influence on grizzly bear activities?
2. Is there a machine learning method that can do all of the following at the same time?
   (a) Analyze a combination of patterns, statistically prevalent and rare, all at once?
   (b) Can the method be easily modified to exclude the common patterns?
   (c) Can the method be easily modified to include only special patterns?
   (d) Can the method be easily modified to include only common patterns?
   (e) Does the method have enough sensitivity to detect specific patterns at different temporal and spatial resolutions?
   (f) Is the method easily modified to include multicriteria similarity evaluations?
   (g) Is the method scalable?
   (h) What are its RAM and secondary storage space requirements?
   (i) What is the expected execution time for different combinations and quantities of patterns to detect, and total volume of data to analyze?

3. Can a modern RDBMS support this type of analysis tools (i.e. efficient and scalable) using only internal resources?

4. Can the method be explained in graphical form to make it more accessible to other developers and researchers?

5. Is there a topological analysis theory capable of explaining all aspects of moving object patterns based on fundamental movement characteristics (e.g. velocity, direction of travel)?

1.5 Research methodology.

To answer the proposed questions, two initial concurrent tasks were undertaken, namely the review of studies of grizzly bears from a natural sciences perspective (i.e. ecological and environmental studies), and the evaluation of different RDBMS. The first, in order to identify the factors that ecologists and biologists have found to be relevant in the determination of grizzly bear behavior. The second, in order to find the most flexible and feature rich RDBMS that can be used for the implementation of the analysis tools under one budgetary restriction, it had to be open source or provided free of charge from the manufacturer.

Subsequently, basic concepts in the computer sciences literature were considered to determine which research results were more relevant to the minimalistic standard ANSI / ISO / IEC SQL implementation philosophy adopted for this work [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. This process further informed the selection of the RDBMS, and identified particular limitations and technical areas that had to be avoided, or compensated for, to ensure a successful implementation. Additionally, the same process identified programming tasks in which the RDBMS excels and that should be taken advantage of to produce better analysis tools.

Next, identification of the specific technical characteristics of the candidate computational methods were examined in conjunction with the details of the tracking dataset to determine their compatibility. In particular, specific tests and statistical measures were applied to the dataset to determine this compatibility. Another benefit of this dataset examination allowed the exclusion of incomplete and inconsistent data that would not contribute to finding substantial answers to the research questions (see subsection 3.3 for a detailed description of these inconsistencies).

Once the candidate methods were selected, the initial implementation began by selecting a small portion of the full dataset to serve as calibration data. After the initial programming and unit testing of the methods, the calibration dataset was processed multiple times to find portions of the programs that could be improved in terms of execution time or secondary storage space utilization.

When the calibration tests were finished, the adequate methods were applied to the full dataset to examine all aspects of its application. The level of success of each method was assessed before full sensitivity, multi-scale analysis, and comprehensive performance analysis were conducted. More details of this process will be presented along with the different implemented methods.

Finally, the results from the methods were examined to determine their relevance, usefulness and novelty, in contrast to the characteristics expected of a successful computational method.

1.6 Scope and limitations.

As stated in the above objectives and questions (1.3 and 1.4 respectively), the current implementation was limited to the internal resources of the selected RDBMS system. However, any and all resources, including advanced methods and intricate administration strategies native to the database management system were employed in order to increase the performance of the adopted solution. To be more precise, any and all software tools made available by the original manufacturer inside or as part of the RDBMS system itself, were considered as an internal resource. This definition excluded all function and data structure libraries external to the RDBMS system that have to make direct use of operating system resources, or that need special modules or agents (such as a listener or similar interface tasks) to interact with other internal parts of the database.

Given the considerable amount of resources necessary to design, construct and test any embedded system, in this case, the tracking GPS collar for grizzly bears, the current work did not develop any part of this important component of the tracking system. Instead, this research focused its efforts on answering the research questions from a comprehensive analysis of an existing grizzly bear tracking dataset.

To fulfill the comprehensive requirement, this research employed fundamental analysis of the tracking dataset from several perspectives (e.g. statistical, topological), as well as applied methods for automated pattern detection.

Any other limitations imposed during the project development were a direct result of the analysis, and will be clearly stated as they are presented to the reader.

1.7 Thesis organization.

Chapter 2 contains a chronological review of literature relevant to this project, divided in two main approaches. Section 2.1 includes environmental studies of grizzly bears in an effort to identify relevant factors that determine their habitat use and behavior. The rest of the chapter includes past studies that analyze and propose solutions to technical aspects of the analysis of moving object datasets in general, and grizzly bear studies in particular (when available).

Chapter 3 includes a non-chronological description of the individual tasks conducted to process this project's dataset and implement its machine learning methods. Important algorithms and programming details will be presented in the text, but full source code lists will not be included in this document. For full source code review, the reader is referred to the accompanying digital media or the project wiki [Specific page to be defined at this time].

Chapter 4 presents the results from each one of the fully implemented methods, and highlights some of the positive and negative characteristics of the resulting programs. In each case, additional limitations and their reasons are included as part of the examination of each method. The results include the main memory and secondary space requirements, performance, scalability, and sensitivity characteristics of the methods.

Finally, chapter 5 contains concise lists of the answers to the research questions, along with brief discussions of their limitations, if any. Also in this chapter, specific scientific and technical contributions are listed, as well as selected specialized areas that can be further examined as a direct result of the conclusions of this work (in an effort to find additional improvements). In other words, that chapter includes the conclusions derived from this research, its contributions (section 5.2) and its future work (section 5.3).

Additional supplementary material is included in the appendices. The reasons for each of them, and a brief description of their contents, will be presented in the text at the point where the work generated that material. Except for appendix **??**, which constitutes a summary of the actual working hours that were employed to bring this research to its current state of completion.

1.8 Chapter summary.

This chapter has presented a detailed definition of the reasons for scientific understanding of movement patterns and geographic preferences for grizzly bears. Its argument comprises the major aspects of the problem definition, which include the increasing volume of tracking data, the minimization of subjective analysis values, and the efficiency of the analysis itself (execution time and statistical relevance). The objectives and research questions were formulated in parallel towards five main goals, the integration of multiple environmental factors for the automated analysis, the desired technical characteristics of the resulting algorithms, the relevance of the algorithm's results in statistical terms, the computational restrictions of the adopted implementation philosophy (i.e. its minimalistic approach), and the existence of a theoretical framework to facilitate topological analysis of fundamental aspects of moving object patterns.

From the software engineering point of view, this is the equivalent of a competing multi objective problem [27, 68, 100, 102, 252], where all aspects have to be carefully balanced in order to arrive at a reasonable compromise between thoroughness and practicality, to produce the most convenient and efficient solution possible.

Chapter 2

Literature review

The purpose of this chapter is to identify important environmental factors that determine movement patterns for grizzly bears, limitations and features of the hardware and software intended for this analysis, conceptual aspects of suitable computer-based analysis methods, and theoretical findings that can assist the current work to answer its research questions.

One of the basic principles of science is to take any problem and divide it into increasingly smaller problems [192]. This process is iterative, and its main goal is to arrive at fundamental aspects of the original problem which should be easier to analyze. After the study of each part, the initial sub-division process is reversed into an integration effort and all of the insights of the smaller aspects are used to explain the original phenomenon [192]. For many problems this has worked remarkably well, but for highly complex problems where all the individual parts have intricate interdependencies, and especially for those problems that have a geographical component, more and more studies are finding that this approach is not the best [45]. Modern studies of wildlife, including studies on grizzly bears, have to include more factors and more interactions amongst them, otherwise the practical application of their results can be severely limited [44, 45, 50].

The understanding of the dynamics of the grizzly bear population in western Alberta, including their habitat preferences and movement patterns, requires a multi-disciplinary approach [195, 204].

One consequence of the increased complexity in the analysis of grizzly bear habitat and behavior data is that the technological tools needed to acquire, represent, store, manage, analyze and manipulate the associated data had to evolve with the studies, and become more complex themselves. The great majority of these changes however, have been introduced in the computer science literature, with little or no effort given to explain the behavior of wildlife.

The rest of this chapter examines aspects of this complex problem from two main points of view. First, section 2.1 reviews studies on grizzly bears from an environmental perspective to identify relevant factors to explain behavior, habitat selection, and movement patterns. Then, the rest of the sections in the chapter analyze technical challenges and solutions to similar problems from the computer sciences perspective.

Section 2.2 presents a brief description of the historical evolution of Database Management System (DBMS) in an effort to identify that technology as a mature and viable solution platform for complex analysis tasks of very large datasets. Section 2.3 presents the knowledge discovery methodology as a comprehensive analysis process for large datasets, making it a suitable alternative for the objectives of this work.

As exploratory steps, this work considered analysis algorithms from the Location-Based Services (LBS) specialty as well as a simulated annealing solution. Although their fully integrated implementation was successful, their scalability limitations characterized them as unsuitable for the analysis or the current grizzly bear tracking dataset. The material is included as additional support of the maturity of the DBMS. The observed details in theoretical, implementation and initial result terms are included in appendix A.

Section 2.4 presents details and previous application publications of the local alignment method, that after the exploration mentioned in the previous paragraph, seemed better suited to handle the analysis of the grizzly dataset.

The material originally destined for section 2.5, and ultimately diverted to appendix B because of their limitations, contains several theoretical analysis frameworks for moving objects. Finally, section 2.6 presents a brief summary of this chapter.

2.1 Grizzly bear tracking, habitat, and behavior.

This section is a summary of studies on grizzly bears from an environmental point of view. Each subsection groups studies based on the main innovations that each one contributed to the understanding of the bears' habitat and behavior, starting with the improvements of tracking and habitat mapping methods.

2.1.1 Tracking and habitat mapping studies.

The early attempts at scientific study of grizzly bear habitat were very subjective and lacked large area coverage, almost to the point of being nearly anecdotal accounts (e.g. [75]). But the researchers themselves recognized those shortcomings and tried to compensate for them by incorporating more detailed data with larger geographical coverage (i.e. satellite data, [74, 76]). As satellite technology improved and the scientific community gained access to a wider range of satellite images, more formal studies of grizzly bear habitat were undertaken. Since the technology was new, there were still many unknowns about the correct way to use it, specifically about how to classify different measurements in the image into unique terrain features (e.g. rocks, barren soil, urban areas) or plant species. Part of the problem was that the spatial resolution of the images was still very coarse (79 m for Landsat-2, [240]). As technology improved, more sophisticated classification strategies were introduced that used more data sources (e.g. additional spectral bands), improving accuracy and reliability of large scale maps of grizzly bear habitat (e.g. [110]). At the same time, as old satellites were gradually replaced by newer and improved versions with increased spatial resolution and more spectral bands captured simultaneously, the data available to researchers was made more convenient by including georeferencing and corrections of many kinds that the users themselves had to apply in the past (e.g. radiometric, noise and atmospheric corrections). Some recent efforts using remotely sensed data are aimed at detecting habitat modifications by analyzing time-series images of the same geographical area (e.g. [203]).

Additional studies in habitat identification have focused on different aspects. Beier and Noss [38] conducted a review of studies on habitat corridors. Their findings were that generalization of benefits to all species is rather difficult because, each species and geographical area has particular needs and characteristics that have to be evaluated and tested for. However, they also found that the better-designed studies (those that recognized, accounted and corrected for their own limitations) strongly suggest that corridors are valuable for ecosystem conservation efforts. The researchers remind decision makers in charge of administering resources for wildlife and humans alike that any effort to change the use of current wildlife habitat should first conduct proper scientific studies in order to prove that said changes will not affect any target species negatively.

Concurrent with the improvements in habitat mapping and identification, observation and recording methods for the actual movements of the bears went from individual bear tracking on foot, passing to capture and tagging methods (radio telemetry), to the current dominant capture and collaring method using Global Positioning System instruments. These changes increased the amount and accuracy of the movement data available to such an extent that, its analysis required a shift from manual log examinations to computer-based methods.

2.1.2 Habitat, movement and activities.

Modern studies of the movement and activities of grizzly bears have found different determining factors for specific behaviors.

Gibeau [114] found that female bears preferred certain areas out of habit, ignoring nearby higher quality areas. The study also found that bears residing in regions with restricted human access consistently used higher quality areas compared to bears where human activities were unrestricted.

In a study by Munro et al. [183], the activities of grizzly bears in west-central Alberta were analyzed in direct comparison to available seasonal food sources in order to determine the importance that said sources have on the activity selection of the bears. The study did not find major differences in the use of specific food sources for this population as compared to other grizzly populations in the central Rocky mountains. It also found that activity selection is directly dependent on local habitats and time of day (e.g. bedding occurred predominantly in forested areas and at night). Significant local variations were explained in terms of a segmentation of the population of bears based on elevation of their core home ranges. Mountain bears were defined as those for which 80% or more of their core home range was above an elevation of 1,700 m, while the rest of the animals were categorized as Foothills bears.

Excluding conventional habitat quality measurements (i.e. subjective habitat quality and connectedness), and using a Resource Selection Function (RSF) [46] and a Step Selection Function (SSF) [217], Chetkiewicz [57] analyzed GPS data to identify movement corridors. Results indicate seasonal variations of said corridors according to food availability.

All these examples point to the fact that habitat preferences and movement patterns for grizzly bears are influenced by many factors such as intensity of human activity, available resources (i.e. food and water), season, age, gender, presence of offspring, geography of the terrain, and time of day, just to name a few.

2.1.3 Tracking with GPS instruments.

Even ten years ago, the technology for wildlife tracking was not developed enough to provide the level of detail that is apparently needed to analyze this problem [114].

However, through the efforts of many scientists in this field, researchers now have some highly detailed datasets that can be analyzed in different ways. In the research by Hunter [139], a new instrument was developed and deployed to take advantage of new technologies such as accelerometers to acquire a very detailed dataset of grizzly bear movement. Carra [52], analyzes a dataset of GPS points for five grizzly bears acquired at 20 minute intervals in which scale effects are significant. In both of these studies practical considerations are identified as limiting factors in the use of the new devices, e.g. the cost of capturing large carnivores, the cost of developing the devices themselves, the mass and volume of the collars, the balance between the amount of data to be captured and the life of the batteries, etc.

Frair et al. [109] analyzed trial generated GPS location data in conjunction with an RSF in order to find and correct statistical biases, such as position acquisition failures due to terrain and vegetation obscuring of satellite signals, and GPS equipment orientation. Their results indicate that it is possible to generate a mathematical error model in order to minimize the errors introduced by environmental and equipment factors. They also conclude that the error model used has to be designed specifically for each collar model (including sampling characteristics), the wildlife species studied, and the land cover characteristics of the geographical area in which it lives.

As Heard et al. [130] found, because of the technical limitations of GPS tracking equipment (i.e. position fix failures), an additional factor to take into account in this type of study is the possible underrepresentation of the use of dense canopy cover sites. A possible correction for this problem is the improvement of the satellite constellation available, and the tracking equipment itself (hardware and software). Both of these require a massive amount of effort, both financially and in man-hours, and cannot be easily undertaken by a single study; rather, the benefits of gradual technological advancement have to be adopted as they become available. In the meantime, the software tools, statistical or otherwise, selected to analyze these datasets have to explicitly address these difficulties as well as examine the possible presence of autocorrelation (positive or negative, [108]) in order to improve the quality of their results.

2.1.4 Mathematical models.

A separate, but closely related line of studies, consists of integrating all of the previous findings on the subject of grizzly bear behavior (i.e. movement tracking) and habitat preference in order to generate computer and mathematical models as tools for land managers, conservation organizations and policy makers in an effort to inform them of the best strategies to balance the human and the bear needs.

The goal of Nielsen [188] was to generate a two-dimensional model to predict the population viability of grizzly bears in west-central Alberta, Canada. Using GPS tracking data and geographic methods, the research integrated seasonal food sources, human-caused mortality, competition for resources, genetic factors, segmentation by gender and age, and habitat loss to generate a model used to predict the density and viability of the bears in a 100 year period. One of the conclusions of the study states that grizzly bears currently use food sources in human-created forest clearings (clearcuts) because of their lack of risk indicators. The research also concludes that despite a 10% population increase factor (hardcoded in the simulation software), by the 30 year mark, a substantial decrease of the foothills population is evident, while at the end of the 100 years simulation, the safe territory areas had been reduced between 54% and 67%.

Negative interactions with grizzly bears increase in frequency as human expansion increases, forcing the bears to move into lower quality habitats and increasing their mortality rates [39, 40, 58, 190].

In the province of Alberta, outside of national and provincial parks, most of the interactions between grizzly bears and humans take place on agricultural lands. To better understand the presence of grizzly bears in those areas, Collingwood el al. [70] conducted a study in which they devised three new methods for classifying Landsat-5 Thematic Mapper images in order to identify specific crops. The bands used were one through five and seven at 30 meter spatial resolution. Their work improved on previous remote sensing data processing methods because the older techniques only returned generic classes for all herbaceous and agricultural areas. Once the new crop maps were generated and validated, the researchers used grizzly bear GPS positioning data to identify the areas the bears visited with the highest frequency. The study found that the bears' presence was mostly during the summer months, 77% of the time was in grass/forage areas (i.e. grass or crop residue) and the rest of the time in small grain areas or unplanted fields (i.e. small grains or fallow fields).

Integrating habitat and human disturbance factors at the same time, Apps et al. [24] conducted a study in British Columbia, Canada, where grizzly bear presence was established with hair-trap sampling and Nuclear Deoxyribonucleic Acid (nDNA) analysis. The researchers conducted extensive statistical analysis at various spatial scales to test the correlation between the density and distribution of grizzly bears and variables reflecting terrain, vegetation, land cover and human influence. Their overall goal was to generate a model to predict multi-year distribution and abundance of the bears in their extended study area. Their general conclusion integrates the fact that for the past two hundred years human activity has forced the grizzly bear population in British Columbia to move towards areas with steeper slopes and more rugged terrain. Comparing these results with the study by McLellan and Hovey [173], where high value habitat areas are present and preferred by bears, but have no permanent population centers and require more than two hours of travel on unpaved roads to access, their recommendation to wildlife managers is clear, the intensity of human activity is a major negative factor (i.e. bears will endure harsher conditions in order to avoid humans).

2.1.5 Conservation.

From a conservation focused perspective, the study by Molitor [180] measures the effectiveness of information programs for back country visitors designed to avoid unnecessary human confrontations with grizzly bears. The research was conducted at the Flathead's National Forest, Jewel Basin Hiking Area, Northeast Montana, USA (on the USA border with Canada). The objective was to identify which of several message-persuasion programs being introduced was most effective in educating visitors to adopt safe practices (e.g. make appropriate noise while hiking, hang food) to avoid negative encounters with bears. The study established test and control groups and found that all written messages delivered directly to visitors were of almost equivalent effectiveness.

2.2 Overview of the evolution of databases and database servers.

This section is primarily based on my fifteen years of professional experience in the industry of database administration and development. The section considers some of the major changes in relational database technologies looking to advance the notion that such systems represent a mature enough platform for fully integrated implementation of complicated analysis tasks. Particularly for implementations of the standard ANSI / ISO / IEC SQL specifications [13, 14, 15, 16, 17, 18, 19, 20, 21, 22] that would allow the same executable code to be ported to other commercial RDBMS systems with minimum changes.

As presented in section 2.1, the rapid increase in the size of the wildlife tracking datasets made manual analysis methods very impractical for researchers, if not impossible. The use of automatic analysis tools was imperative, and the modern electronic computer was perfectly suited for such tasks.

The problem of processing high volumes of wildlife tracking data in a timely manner was approached from very different perspectives, and was concurrent with the size increase of other datasets (e.g. enterprise-wide sales data, historical bank transactions). After a few years of their introduction, relational databases established themselves in the industry as the *de facto* standard for storage and management of large volumes of alphanumeric data [234]. Almost at the same time, business managers and company executives started asking for complex business reports that included increasingly larger amounts of detailed data (e.g. several weeks of store-wide item-level sales data). To satisfy these needs, specialized reporting tools were developed and some support structures for those were integrated by the developers of the relational database management systems themselves. Two years or so after that, a further data volume and reporting power expansion demanded even larger amounts of data, presentation methods (e.g. quarterly and annual enterprise-wide reports by state, province, region, sales district), and almost constant processing (e.g. by-the-second stock market quotes, daily enterprise-wide reports). These new demands on storage space and performance revealed important limitations of the existing class of relational databases. A new type of relational database had to be created to handle these special requirements, the data warehouse [247].

To make a proper distinction between the two different kinds of databases that a single company could have, the initial databases that were designed to take care of the day-to-day operations of the company were defined as transactional. Making direct reference to their function as the tool with which each individual transaction in the entire company was registered, tracked and resolved in case of any conflicts (i.e. every item sold, every bill paid, every employee hired). While the newer databases designed to support the generation of complex reports from the company-wide datasets were named data warehouses.

As the name implies, a data warehouse is a repository of very large amounts of old company transactions whose primary tasks consist of the production of reports and data analysis. At the very beginning of their evolution, data warehouses were little more than a copy of the contents of the transactional databases in separated servers, with the sole purpose of not slowing down the operation of the entire company when a complex set of reports needed to be generated. As time passed, and the specialists in charge of maintaining data warehouses (i.e. the database administrator) identified several types of cycles in their use (e.g. bi-weekly payroll cycles, monthly inventory cycles, quarterly reporting cycles), data warehouses became more specialized and efficient [211].

This separation of functions allowed the independent evolution of both kinds of database platforms.

The transactional databases remained the heart of the daily operation of the company, emphasizing speed and availability of relatively recent and relatively small data elements. The time interval of transactions kept directly in the transactional database mainly depends on five factors.

1. The business model, i.e. how many transactions are generated in the company per unit of time, and how many of these units have to be directly available to allow for the efficient operation of the company.

2. The server's secondary storage capacity, i.e. the size of its hard drive arrays.

3. The level of acceptable risk of loosing any amount of data before a copy is made into an independent machine (data warehouse or otherwise).

4. The speed and capacity of off-server secondary and tertiary storage platforms (data warehouses and backup systems such as tape units).

5. The legal regulations concerning the time span of operational data that each company has to have instantly available at all times.

There are two main approaches to satisfy the demand of the modern transactional database server, the monolithic and the distributed server [212].

The first type is a very large machine with many hot-swappable modular redundancies built-in, such as Central Processing Units (CPUs), RAM memory banks, on-the-fly replicated hard drive units (RAID 5 is the industry standard), and power sources. Some of the most sophisticated (DBMSs) support this kind of server by allowing the concurrent operation of several identical servers with simultaneous replication of transactions. Only one of the servers is the active server, while the others are the standby servers. In the case of a failure of the active server, by a preprogrammed arbitration method, one of the standby servers will automatically become the active server taking over all operations without any disruption of the business user's activities.

The distributed server is a group of smaller machines (as compared to their monolithic counterpart) that work cooperatively to satisfy the demands of their users. As a consequence, each server can be located in a different geographical location if it is connected via a network of sufficient speed. In this type of architecture, a special program called the scheduler is responsible for receiving all the data requests from the users, assign the tasks to individual servers by a load balancing algorithm, and for sending the results back to the user processes [244]. The industry leaders in the DBMS market have included in their software the possibility of having more than one scheduler process active at any given time; in fact it is recommended in their manuals to avoid a single point of failure in this regard [104]. With several schedulers active per server, database administrators are responsible for programming additional load-balancing policies under normal and failure circumstances so that only one scheduler per server acts as the communications link between cooperative servers. This main scheduler process is designated the primary scheduler of the server. To make the operation of the entire system as fast as possible, it is recommended that the main scheduler processes of all servers get connected amongst themselves by special network connections of higher speeds than those used to connect user processes.

The modern servers designed for transactional databases typically have multi core and multi thread processor units*, large amounts of random access memory (64 GB of RAM is common) [199], high-speed network modules (10 Gigabit Ethernet) [197], real time replicated disk storage in a different set of cabinets (1 TB of RAID 5 is typical), hardware integrated encryption support [198], and in some cases, direct hardware structures to support specific manufacturers' solutions (Oracle/Sun Microsystems SPARC data blades with Oracle Solaris Containers' support) [196].

Returning for a moment to the evolution of data warehouses, their first operational improvement originated from the level of aggregation that the most common types of reports required, i.e. if the most commonly generated report included regional level sales data, the most detailed level of data stored in the data warehouse was at the regional level. For those early implementations, the data warehouse read from the transactional databases (always after hours), and aggregated sales data by store, region, etc., before writing it to its repository. Later improvements were introduced to make the report generation process more efficient [211]. Modern data warehouse servers are slower compared with their transactional counterparts, typically having only half the RAM and half the cores and threads, but with many times the secondary storage capacity (100 TB of non-RAID disk space is common), and practically unlimited tertiary storage (i.e. tape units). The reporting software uses the additional secondary storage to maintain intermediate and final results instead of making heavy use of the server's main memory.

After approximately three to five years of deployment, data warehouses became so large, and included data from such heterogeneous sources that, traditional reporting methods proved to be ineffective in finding new facts about the companies [211, 212, 234]. One of the reasons for this limitation turned out to be the parametric formulation of the reporting tools, i.e. a reporting program will only include data for which it has its values

*To support large quantities of concurrent users and to improve multitasking speeds.

directly included in its program, ignoring any other values. To minimize this limitation, machine learning and data mining methods were created.

2.3 Knowledge discovery and data mining.

As Han and Kamber point out in their text [128], strictly speaking, data mining is just one of the steps necessary to analyze large amounts of data in order to extract valuable knowledge. Therefore, the more comprehensive process of analyzing large datasets in order to extract valuable knowledge, is defined as Knowledge Discovery in Databases (KDD).

Specifics of the KDD process differ from one author to another, but in general terms they all agree that some level of interpretation has to be included in order to extract real value from the facts uncovered by the computer methods alone. Fayyad et al. [101], Qi and Zhu [206], and Miller and Han [179] consider the acquisition of expert knowledge in the domain of interest an integral part of the process, and in order to minimize its subjectivity, include a first step, "Framework integration". Adding step number zero (0) to the methodology of Han and Kamber, produces the following set of steps for the KDD process (paraphrasing and combining from all the texts cited so far in this subsection):

0. Framework integration. To acquire an expert background in the domain of interest.

1. Data cleaning. To remove noise and inconsistent data.

2. Data integration. Where multiple data sources may be combined, usually in a data warehouse.

3. Data selection. Where data relevant to the analysis task is retrieved from the database.

4. Data transformation. Where data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.

5. Data mining. Where intelligent methods are applied in order to extract data patterns.

6. Pattern evaluation. Where the experts in the subject make educated interpretations, to identify the truly interesting patterns representing knowledge based on some objective measure. This step is where information is transformed into knowledge.

7. Knowledge presentation. Where visualization and knowledge representation techniques are used to present the mined knowledge to the user.

This integrated and more comprehensive approach will be used during this work.

Going back to the data mining process, it can be described as a computer program designed to analyze large amounts of data based on one or more mathematical measurements (e.g. correlation, entropy, information contents, geometrical similarity) in order to extract patterns [179]. Han and Kamber [128] define data mining methods to be of three basic types, data classification, clustering, and time series characterization, each type is defined as follows:

Data classification methods generate a relatively small number of groups of data (i.e. the classes) in which variation within the group is minimized, while variation between groups is maximized [128].

The definition of clustering methods is very similar to the classification methods. The main difference is that clustering methods involves geometric or geographic objects (e.g. polylines, polygons), while classification methods are almost always applied to abstract data (e.g. reflectance values, hysteresis thresholds) [128].

The time series characterization methods can be used with any kind of data as long as a time component is present. The time component will then be used to divide the data in smaller subsets. These methods are used to identify changes or similarities in the subsets from one point in time to another.

Based on the characteristics of the data mining methods and the size of the modern wildlife tracking datasets, data mining appears to be well suited to analyze the latter in order to discover new knowledge.

2.4 Pattern identification with data mining methods.

One of the fundamental concepts in all data mining methods applied to geographic applications consists of analyzing a set of trajectories to identify the longest common subsequence amongst them. Translated into computer tasks, there are two different approaches to find these common subsequences, exact matches and approximate matches. In general, the exact matches are faster than their approximate counterparts, but they are also more restricted because the typical exact matching method has to search for smaller common subsections of trajectories (e.g. small numbers of consecutive positions from different trajectories that are at a given maximum distance from each other).

Relaxing the geographic requirements of distance, i.e. momentarily ignoring the “near” and “far” thresholds, and converting the consecutive trajectory positions into concatenations of abstract symbols of an alphabet, these trajectory subsequences can be seen as small sections of text (a.k.a. patterns, sequences, etc.) as compared to a much larger databases of full texts (e.g. the database of the Human Genome Project, the entire works of William Shakespeare, etc). With this re-definition in mind, many other scientific areas have produced algorithms to solve specialized versions of this same problem applied to their respective specialties.

According to the most recent version of the taxonomy proposed by Cleophas et al. [65], almost all of these solutions can be related to each other according to their specific technical details, the order in which the details are applied, the type of data structures they use, and the mechanisms with which the methods manipulate their data structures. The main goals of the taxonomy are to provide a classification tool to facilitate the comparison between specific implementations, and to serve as a didactic tool. The algorithm comparisons included structural and performance aspects, while the didactic goals have the added benefit of being able to automatically generate many variations of the same main algorithm for additional studies (computational complexity, completeness, correctness, main memory requirements, secondary storage space requirements, and performance) [65].

Earlier work on the subject by Cleophas [63] started with the mathematical analysis of some classic pattern matching algorithms, emphasizing the relevance and equivalence of pattern matching trees, finite state automatons, and regular grammars, until the main branches of the taxonomy could be identified [63]. In subsequent work, also by Cleophas [64], additional refinements to the algorithm analysis methodology were included, and the taxonomy structure was refined. Many more algorithms originating from academic and empirical domains were mentioned as candidates for analysis (e.g. Aho-Ganapathi, Aho-Ganapathi-Tjiang, Cole-Hariharan, Commentz-Walter, Dubiner-Galil-Magen, Hoffmann-ODonnell, Kosaraju, Watson). But the author reminded the reader that, in contrast to biological taxonomies that are defined by ancestry, this algorithm taxonomy is driven by functionality details, and that the specific order in which every scientist decides to introduce each detail might produce a different taxonomy. In other words, that there is still a subjective factor in the production of the proposed algorithm taxonomy, and that there is still more research to conduct in that regard to minimize or eliminate that subjectivity.

The main conclusion of the taxonomy studies could be interpreted to be the relevance that pattern matching algorithms have in academic and empirical terms, giving emphasis to the understanding of their parts, variations and efficiencies (or lack thereof). A secondary conclusion, directly derived from the taxonomy itself, can also be identified as the recognition of two main families of algorithms, the exact matching group and the approximate matching group. The work by this group of researches was focused mainly on the exact matching algorithms.

Two independent reviews on approximate matching methods can be found in [185] and [49]. The first review, by Navarro [185], focused exclusively on the online processing of long texts compared to relatively short patterns, and only on the matching methods that allow for a certain number of errors (i.e. approximate matching). It explained the basic mathematical and practical concepts of the problem, some of the application areas, and compared the main versions of each solution with empirical measurements. To explain each one, the document divided the solutions into dynamic programming, algorithms based on deterministic automata, bit-parallelism techniques, and filtering algorithms [185]. The first two solutions being the oldest and most fundamental to the problem, while the later two are more contemporary refinements to those.

The review by Boytsov [49] presented approximate search algorithms divided into two major categories, direct and sequence-based filtering methods. The study focused on retrieval operations of infrequently updated dictionaries for natural English and Russian (synthetic and extracted from a large database), and Deoxyribonucleic Acid (DNA) sequences. The author defined the direct methods as algorithms looking for complete patterns, and included all theoretical and implementation details of prefix trees, neighborhood generation, and metric space search methods. Sequence-based filtering methods were defined in operational form as those having at least one filtering step and one checking step. The main characteristics of the filtering methods being the computation of the edit distance between the search pattern and a candidate string, and their filtering efficiency, which is a measure of how many dictionary strings are discarded in the filtering operation [49]. From those definitions it followed that the direct searching methods, implemented with exact searching algorithms, are more appropriate for the identification of patterns from a dictionary set as compared to candidate strings (or texts).

With an exact matching philosophy as their central point, subsections A.1, A.2 and 2.4.1 present methods originating from the mathematical optimization, telecommunications and the biotechnology specialties, respectively.

2.4.1 Sequence alignment.

Using the basic principles defined by Hamming for binary alphabets [126, 127, 181], Levenshtein [163] studied the mathematical characteristics of all binary codes capable of detecting and correcting word transformations consisting of insertions, deletions and reversals. The objective of the work was to determine how many additional characters per word would have to be added on one end of a telecommunication's channel, in order to detect and correct a given number of errors at the other end, introduced by the transmission media. In other words, create an automated process that can uniquely identify all the individual changes needed to transform one binary word into a second word, not necessarily of the same length. Figure 2.1 shows two possible sequence alignments between one pattern subsequence and one text section. Which of the possible alignments is correct, depends on the objectives of each process.

Independently of those efforts, Needleman and Wunsch [187] defined a method of global alignment for two amino acid sequences that proposed the minimum number of cumulative edits to one of the words until it was transformed into the second. The alphabet in this case consisted of the twenty different amino acid residue characters (a.k.a. residues, amino acid elements, amino acid symbols, subsequence characters, symbols, etc.), and the individual character changes included insertion, deletion and replacement of one character by another. The method defined the concept of minimum edit distance as the smallest sum of all single-character changes to achieve the full transformation from one word into the other, and vice versa. An important theoretical detail of this method was developed by Huang [138], who identified the execution time of this solution to be of order $O(n^2)$.

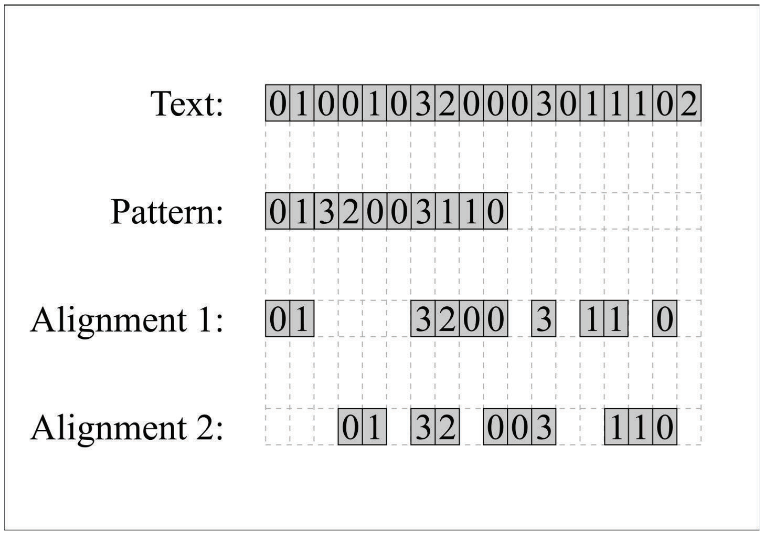

Figure 2.1: Sequence alignment.

A few years later, and almost unaware of the implication, Sellers [222] unified both theories by demonstrating that the changes in Levenshtein's method were exactly the same as the edit distance in the work by Needleman and Wunsch, only applied to different alphabets.

2.4.1.1 Local alignment.

To reduce the amount of work, as compared to global alignment, local alignment methods look for exact matches of very small subsequences throughout the larger symbol sequences. Early work on the subject by Morrison [182] established execution improvements applied to retrieval of short text sections from large flat files of alphanumerical data by utilizing an auxiliary indexing structure. The study provided specific algorithms to build, maintain and use the indexing structure to find all occurrences in the flat file of a single keyword. Intrinsically, the work used the concepts of prefixes and suffixes to build the index and to speed up the keyword searches.

Subsequent work on the subject by Boyer and Moore [47, 48] produced one of the first semi-linear execution time algorithms. The method explicitly used the concepts of proper suffixes, and the counter intuitive notion of exploring individual character matches starting from the extreme right side of the pattern. This "backwards" exploration was justified by the idea that the farther right a mismatch of text and pattern occurs, the larger the possible displacement of the pattern in relation to the text. This displacement, or slide as it is commonly referred to, would be limited only by the proper suffixes of the characters previously matched from the pattern that exactly repeat farther left in the pattern itself. The authors also explored the implications of such facts, finding boundary values for the best and worst execution cases, stating that the length of the searched pattern and the number of symbols in the utilized alphabet were determining factors in the performance of the method. Somewhat surprisingly, the authors also determined that the structure and contents of the text do not represent determining factors for the performance of the algorithm.

An alternative sub-linear algorithm, independently proposed by Knuth, Morris and Pratt [147] utilized virtually the same ideas as the Boyer-Moore algorithm, but instead of suffixes, used the opposite concept of prefixes, and examined the text and the pattern starting from their left-hand end. The work also presented several empirical improvements aimed at increasing the efficiency of the average cases (mid-size patterns and reasonably large alphabets), and provided possible modifications to the Boyer-Moore algorithm to make it faster. Those ideas, along with further refinements, were used by the same authors in a later version of their algorithm [148].

Developed at the same time as the Boyer-Moore algorithm, the work of Aho and Corasick [1] explored possible solutions to the problem of finding all occurrences of a set of different patterns, referred as the dictionary set in the study (a.k.a. keyword set, pattern set, etc.), inside very large text blocks. The proposed algorithm employed a preprocessing step that constructed a finite state automata to organize all patterns, and identified the largest common suffixes of each execution path as compared to every other execution path in the conceptual machine. The individual character comparisons of the method followed the straight forward approach of comparing text and patterns starting from their far left end, and was measured to perform in sub-linear time for bibliographic searches.

Figure 2.2 shows the key conceptual structures of the Boyer-Moore (without failure links for the pattern subsequence, explained below) and the Aho-Corasick algorithms, mainly to illustrate the individual character matching direction, and the pattern sliding directions. Please note that the Aho-Corasick concepts are not presented in its original finite state automata formulation, but in its equivalent hierarchical tree representation, in order to make subsequent comparisons between methods easier to explain in this work.

Fundamental to these approaches is the concept of failure links as applied to the dictionary set, which is a mechanism for preserving the maximum amount of already matched characters immediately after a mismatch [124]. To better illustrate this concept let us consider an example. Figure 2.2.b shows the hierarchical tree structure for the dictionary set composed of {020, 0221, 2002, 201, 20221}. Let us imagine that the tree has been traversed with the text characters “202”. For the straightforward approach, if the next character in the text is **not** a 2, a mismatch will occur, the tree will be re-aligned with the text one position to the right, and the matching process would begin again from the root node. In the Boyer-Moore and Aho-Corasick solutions, the pattern data structures contain pointers to other places in the pattern where proper suffixes (shorter sections of common characters) can be found [124].

Returning to the example of the Aho-Corasick approach, after the mismatch is detected (traversing the tree with “202”), the process will transfer the matching (via the appropriate failure link) to the tree branch containing patterns {020, 0221}, the tree will

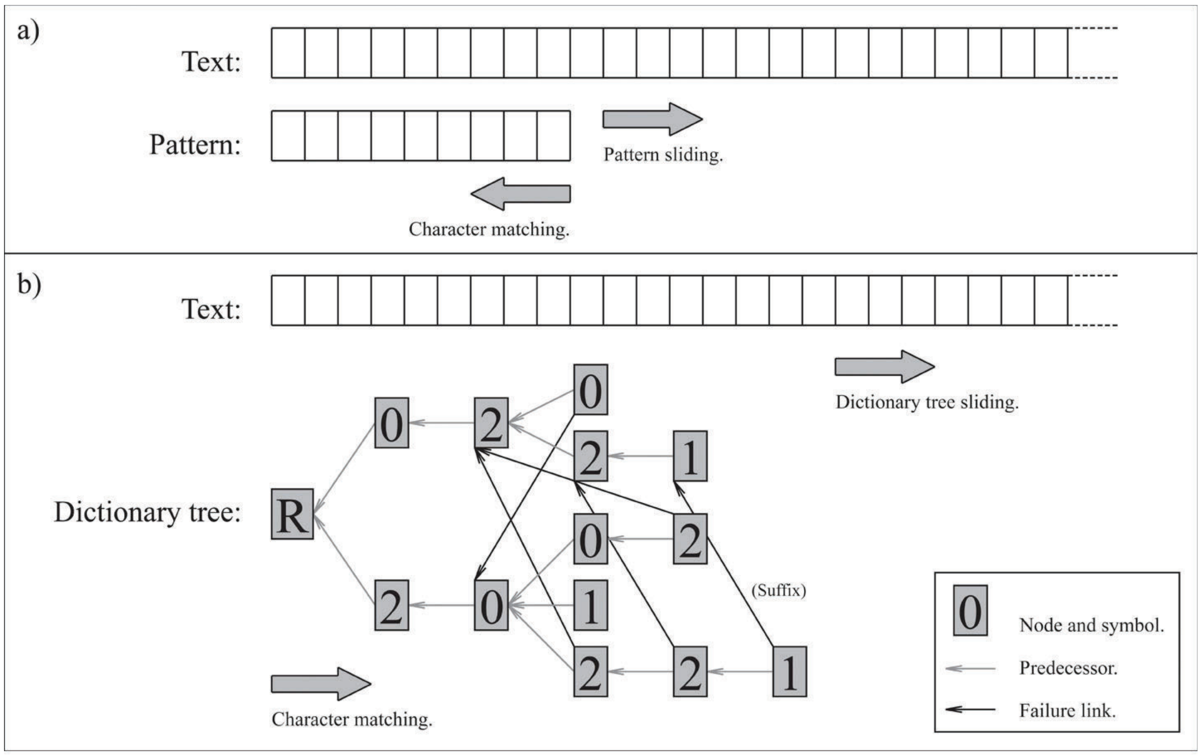

Figure 2.2: Key concepts of two exact string matching algorithms. a) Boyer-Moore (1975, 1977) and b) Aho-Corasick (1975).

be re-aligned with the text one position to the right, and the matching process would resume from the corresponding node as if the text characters "02" had been traversed from the root, i.e. saving the process to have to match the first two characters in the possible matching subsequence.

In essence, the use of failure links in this way minimizes the number of times every character in the text will have to be examined to identify all overlapping pattern matches.

Complementary analyses to the Boyer-Moore approach followed in order to prove its correctness in mathematical terms, and to provide complexity analysis aimed at defining its lower and upper bounds [69, 123, 207, 231].

The work by Horspool, [137], compared the performance of a slightly improved version of this algorithm against the direct use of microprocessor built-in character-searching instructions of some computers of the time, e.g. the "Search Equal" instruction of the

Universal Automatic Computer (UNIVAC) 1100. The major conclusion of the work stated that the Boyer-Moore method outperforms these special-purpose instructions for all cases, except when the searched patterns are very short.

Additional improvements to the Boyer-Moore algorithm were proposed by Apostolico and Giancarlo [23] by observing that, if the search method keeps a running record of the examined characters, in terms of the repetitive suffixes that occur in those characters, the algorithm minimizes the number of times that each text character has to be examined. The method used a linear-time execution method to preprocess the pattern, and had a worst case execution time proportional to twice the number of characters in the text (linear).

2.4.1.2 Local alignment in Biotechnology.

Returning to the objective of detecting common subsequences between two larger symbol sequences, the study of Smith and Waterman [228] proposed a dynamic programming solution that ranked all possible alignments of one full sequence against another, including the option of symbol deletions via a statistically quantified "deletion weight" value (their major contribution as compared to the Needleman and Wunsch method described above). The work employed a bidimensional matrix to determine all possible sequence alignments ending at the symbols represented by each matrix cell, and a simple similarity formula, which is the essence of local alignment. The method has a quadratic upper bound, and to be of practical use requires the length of one of the compared sequences to be small in relation to the other(s).

In biotechnology research, the restriction on the size of one of the subsequences is not critical, and the Smith and Waterman algorithm was adapted for many studies (as described in the rest of this subsection). In such studies, the overall objective is to find long DNA sequences (i.e. entire molecule formulas) that contain short functional groups that determine specific physical and/or chemical characteristics (e.g. hydrophobic, hydrophilic, polar). The biotechnologists start with a small amino acid subsequence (four to seven symbols) of a known molecular function, identify all the individual amino acids that can be replaced by a functional equivalent, and generate all possible subsequences with the same number of symbols. The set of equivalent subsequences is then ranked, generally with the use of a pair substitution matrix that reflects the relative abundance (i.e. frequency) of all pairs of amino acids. The highest ranked subsequences are selected, and exact match searches are conducted against entire molecule databases [37].

Another early effort with this overall goal can be found in [167], where an amino acid replacement matrix was defined to give higher rankings to likely evolutive pair replacements when comparing protein subsequences. The search method known as Protein Sequence Alignment Software (FASTP), employed a lookup table to establish positional identities amongst individual residues, and a “diagonal similarity region” technique (as described in [245]) to establish the starting and ending positions of all consecutive subsequence identities. The method was limited because it did not allow direct insertions or deletions of residues at the matching stage, instead it allowed them after the maximum similarity diagonal regions were established, at the final ranking stage. Later improvements to the same method by the same authors [205] produced the computer program known as FASTA, which uses a lookup table and a diagonal similarity search strategy (a.k.a. band alignment) just like its predecessor, but adds three major steps. First, the band alignment was modified to allow a limited number of gaps (16-32) amongst the residues. Second, a translation step was added to allow the comparison of protein subsequences to DNA databases; and third, an empirically based approximate gapped calculation was incorporated for segments containing multiple non-overlapping alignments.

The publication by Altschul et al. [8]* presented a very popular approach to this kind

*re-explained in simplified form in [6].

of functional search, that had the additional objective of finding molecules with multiple non-overlapping subsequence matches (the BLAST method). The publication also provided theoretical statistical details that proved the efficiency of the overall method and the amino acid pair substitution matrix. One of its most relevant results, equation 2.1, defined the relative ranking of any pair of amino acid residues in terms of the relative abundance of the pair itself compared against the abundance of its individual component residues.

$$\boldsymbol{s}_{ij} = \frac{1}{\lambda} \, log \left(\frac{\boldsymbol{q}_{ij}}{\boldsymbol{p}_i \boldsymbol{p}_j} \right) \tag{2.1}$$

Where:

$\boldsymbol{q}_{ij}$ is the target frequency of the pair of amino acids, and

$\boldsymbol{p}_i\boldsymbol{p}_j$ are the frequencies of the individual residues (the pair's background frequency).

Equation 2.1 implicitly shows that the method is limited by the amount of data available to generate the log-likelihood values, however, in the case of biotechnology where there are already millions of sequenced molecules (with thousands of amino acid residues per molecule), this limitation is irrelevant. Additional statistical considerations to prove the relevance of the method were presented in [142], and [143].

This approach is equivalent to the Mutation Probability Matrix (PAM) defined by Dayhoff [77], also known as a log-likelihood matrix. The main difference between the work of Dayhoff and Altschul is that Dayhoff's work was almost exclusively based on empirical notions, while Altschul's included the fundamental mathematical principles that make it relevant. Aware of these facts, the work of Altschul et al. [5] examined the relative sensitivity of the PAM substitution matrices at different evolutionary distances (i.e. statistical frequency-change estimations), and concluded that the validity of the matrices decreases as the evolutionary distance increases (e.g. from 1-PAM to 250-PAMs).

For application in biotechnology, the symmetry of the log-likelihood matrix is irrelevant because the molecular functional groups can appear in the longer molecular formulas in any order and permutation, as long as they are contiguous to each other [77]. To eliminate the possibility of improper use in biotechnology applications, the log-likelihood matrices are generated in symmetric form, i.e. for pairs of residues, each pair is counted twice, accumulating their frequency into the two corresponding symmetrical cells of the matrix. Transforming the lambda factor in equation 2.1 into a general scaling factor for the entire matrix.

Henikoff and Henikoff [131] explored a different formulation of substitution matrix, seeking to improve the results of searches of proteins containing functionally similar subsequences. In direct contrast to all previous definitions of substitution matrices, which only included pairs of amino acids, this work defined the Blocks Substitution Matrix (BLOSUM) as the probability of entire subsequences (of more than two residues) against the combined background frequencies of all the residues in the block. Henikoff's work compared the performance of the Basic Local Alignment Search Tool (BLAST), FASTA and S-W[*] search methods employing BLOSUM and PAM matrices of different evolutionary distances, matched by their respective entropy value. Their main conclusion determined that the BLOSUM matrices improved the quality of results when searching for full functional subsequences as compared to the equivalent pair substitution matrices (i.e. PAM matrices). This work suggests the direct extension of the log-likelihood matrix to subsequences with any number of residues.

This in turns requires the log-likelihood matrices of any dimensionality to contain symmetrical cells for each permutation of the evaluated functional group. The level of symmetry of the matrix is defined by its generation procedure and could include "no symmetry", "forwards-to-backwards symmetry", and "full permutation symmetry". No

[*]The Smith-Waterman method.

symmetry means that the subsequence “678” is considered different from all its possible permutations. Forwards-to-backwards symmetry defines the case in which “678” is only symmetrical to “876”, and full permutation symmetry is defined when “678”, “876”, “687”, “786”, “867”, and “768” are all considered symmetric to each other (which can be considered a further generalization of the forwards-to-backwards symmetry).

Figure 2.3 shows a decomposition of the three dimensional forwards-to-backwards symmetry for abstract functional groups generated with four symbols ($\Sigma = \{6, 7, 8, 9\}$), starting with the full matrix (2.3.a). Figure 2.3.b shows the three symbol subsequences that are a direct equivalent to the diagonal elements of a 2D matrix, while figure 2.3.c expands the diagonal concept to a two dimensional plane containing all the palindromes[*] of the three symbol subsequences. To analyze its symmetry, imagine a uniform vector field perpendicular to the diagonal plane starting at the upper right corner of the matrix (the cell labeled “699”) and pointing towards the lower right corner of the matrix (the cell labeled “996”). Any pair of symmetrical forwards-to-backwards cells will be located along one of the uniform vectors at the same distance to the diagonal plane (figure 2.3.d, e and f).

More recent improvements to this method have been presented by the same group of researchers (e.g. [7, 9, 10]), where past theoretical assumptions and limitations have been fully explored and validated, or modified and expanded in some rare cases (e.g. [248]). In those projects, statistical principles and limitations were examined, and important theoretical and empirical modifications were applied. Particularly to the construction of log-likelihood matrices and to the rapid identification of sequences with multiple non-overlapping subsequence matches [221, 250].

In order to apply only one operational version of the log-likelihood ranking system to the grizzly bear tracking dataset, as described above, a similar rule to establish equiva-

[*]The subsequences that are written exactly the same way forwards and backwards, e.g. “787”.

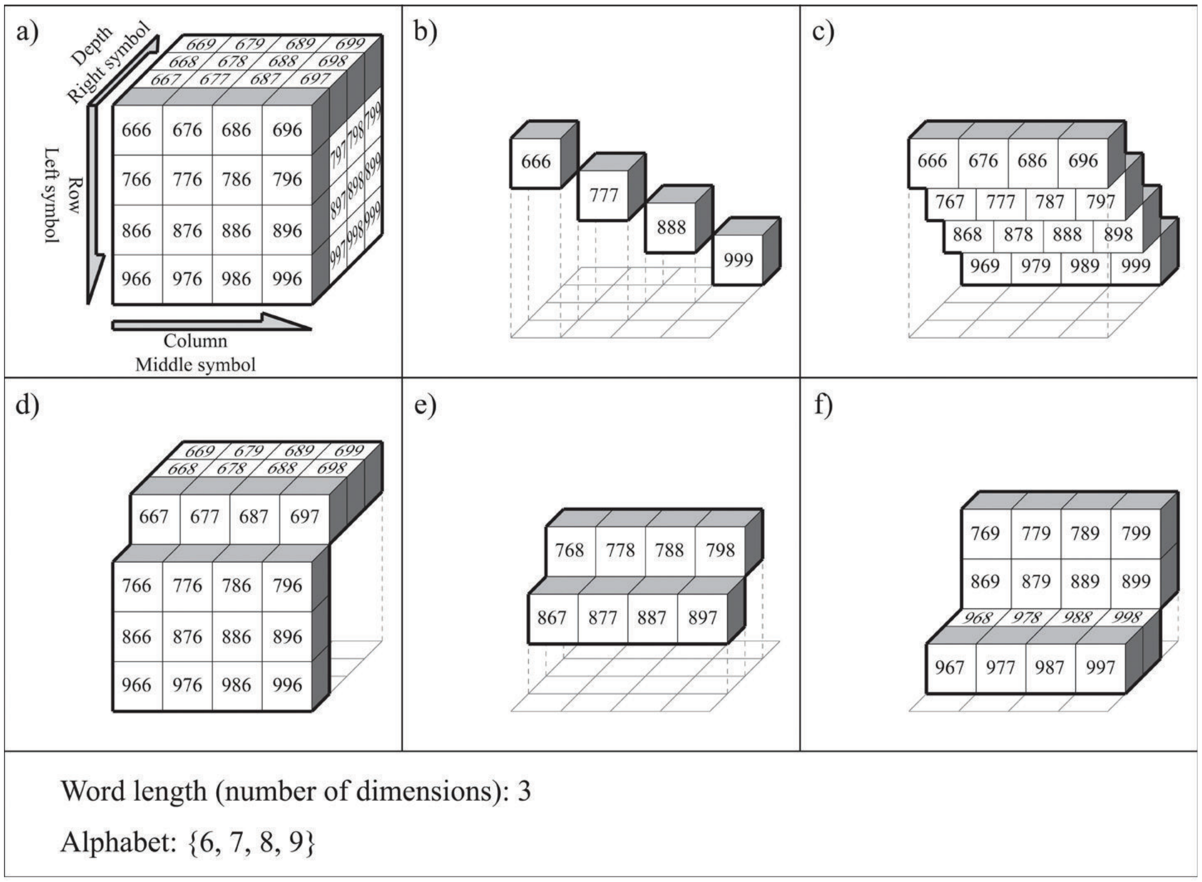

Figure 2.3: Log-likelihood matrix of 3-symbol words, 3D symmetry analysis. a) Full matrix, b) Diagonal, c) Palindromes (including diagonal words), d), e) and f) Forward-s-to-backwards word symmetry.

lence between any symbol subsequence permutations would have to be identified. However, such rule has not yet been defined, then at least two versions of the matrix will have to be applied, an asymmetric and a symmetric version.

2.4.1.3 Keyword trees.

As analyzed in the work by Aho & Corasick [1], the computer representation of a set of patterns to be found in a given text (or set of longer strings) determines the overall performance of the searching operation. Several structures have been designed to optimize specific aspects of the process based on the grammatical characteristics of the dictionary set. Amongst them, prefix trees, postfix trees, trie trees and keyword trees have found the widest acceptance and application [35, 49, 147, 182]. Additional performance gains have also been identified from careful design of abstract data structures manipulated with computer systems [185, 236].

Given the results of Aho & Corasick [1], and the characteristics of the segmentations employed in this work (sec. 3.6.1), keyword trees were determined to be the best candidates for this application.

Without the loss of generality, this analysis continues for the reduced alphabet given by $\Sigma = \{0, 1, 2, 3\}$, and for dictionary sequences with only five characters (shortened to improve clarity in some of the illustrations). Generalization of the results from this analysis can easily be achieved by expanding the alphabet to include additional characters and by increasing the length of the dictionary sequences. Please note that the analysis does not restrict the alphabet of the text itself, only of the dictionary sequences, i.e. the text might contain many more characters than the dictionary sequences, but the valid dictionary words will be comprised exclusively of the characters in Σ.

In regular grammar jargon [3, 4, 103], the alphabet Σ is the set of terminal symbols. Table 2.1 shows the common terminal symbols for the keyword trees.

Table 2.1: Keyword trees' common terminal symbols (alphabet).

Symbol	Interpretation
0	Generic abstract class
1	Generic abstract class
2	Generic abstract class
3	Generic abstract class

For the rest of this work, the terms keyword tree and dictionary tree are used as synonyms.

Once keyword trees were selected, the following step was to analyze the structural and performance behavior of the trees constructed for several sets of dictionary sequences with specific grammatical characteristics. The analysis followed the approach of comparing one character at a time from the pattern and the text, starting from the left end of both strings. The following cases, listed in order of increased specialization, were included:

I. Fully populated keyword tree (F). Where any combination of the alphabet characters is accepted. Fig. 2.4.

II. Uniformly populated keyword tree (U). Where only a subset of alphabet symbols are accepted after a particular character has being chosen. Only {0, 1} after {0, 2} AND only {2, 3} after {1, 3}.[*] Fig. 2.7.

III. Sparsely populated keyword tree (S). Based on the uniformly populated keyword tree, but excluding entire sub-branches. Fig. 2.10.

IV. Distinctively populated keyword tree (D). Where contiguous repetition of the same character is not allowed. Fig. 2.15.

To understand the structural needs of these keyword trees, their full set of failure links had to be determined.

I. Fully populated keyword tree (F).

To improve clarity in the illustrations, figure 2.5 shows only the right-pointing failure links for the fully populated keyword tree (figure 2.4). Figure 2.6 shows the corresponding left-pointing failure links. As expected, all failure links in this version of the tree point to an adjacent branch, and backtrack the process only one tree level.

The individual tree analysis ends with the generation of the production rules of each case [3, 4, 51, 72, 79, 80, 103, 118, 134, 140, 141, 174, 175, 226]. Table 2.2 contains the non-normalized production rules for the generation of the fully populated keyword tree (F).

[*]Which is equivalent to say: “No consecutive ones, and no consecutive twos”.

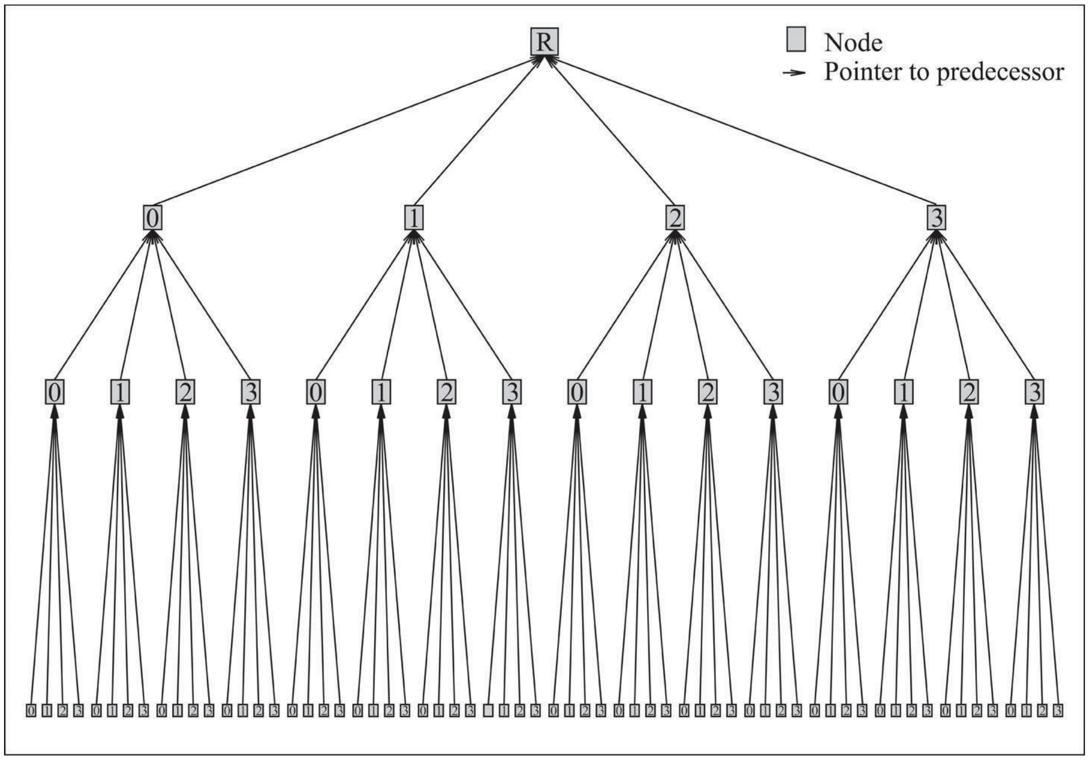

Figure 2.4: Fully populated keyword tree, height = 3.

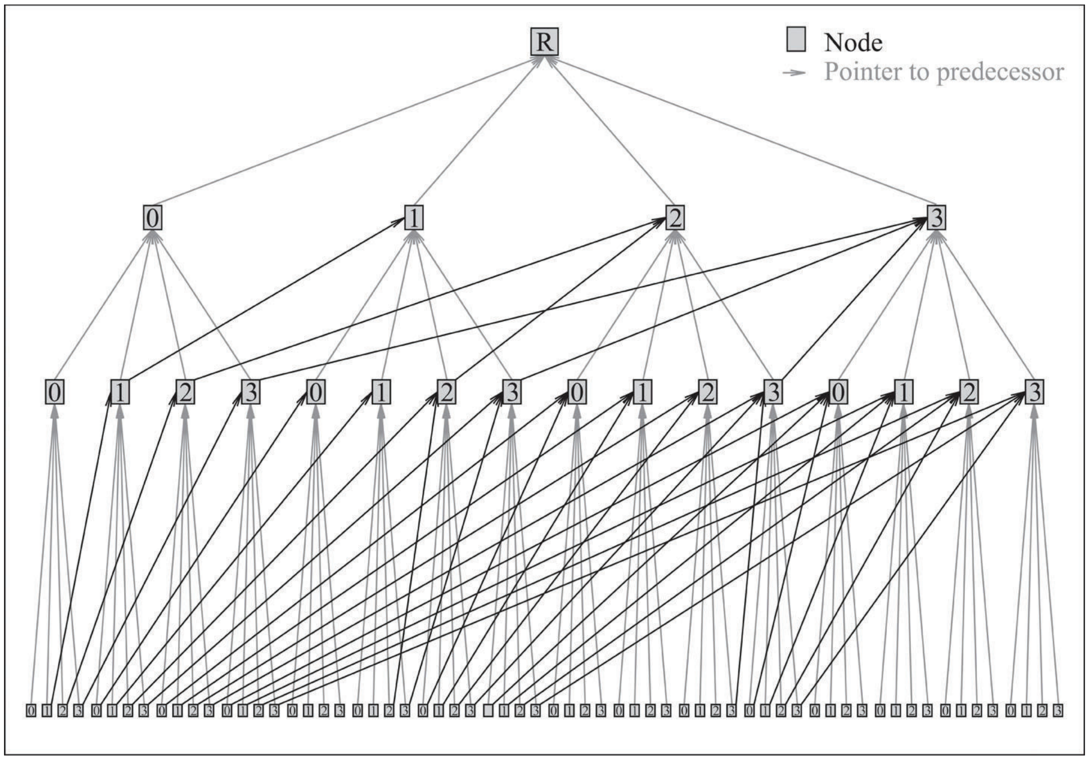

Figure 2.5: Fully populated keyword tree with right-pointing failure links.

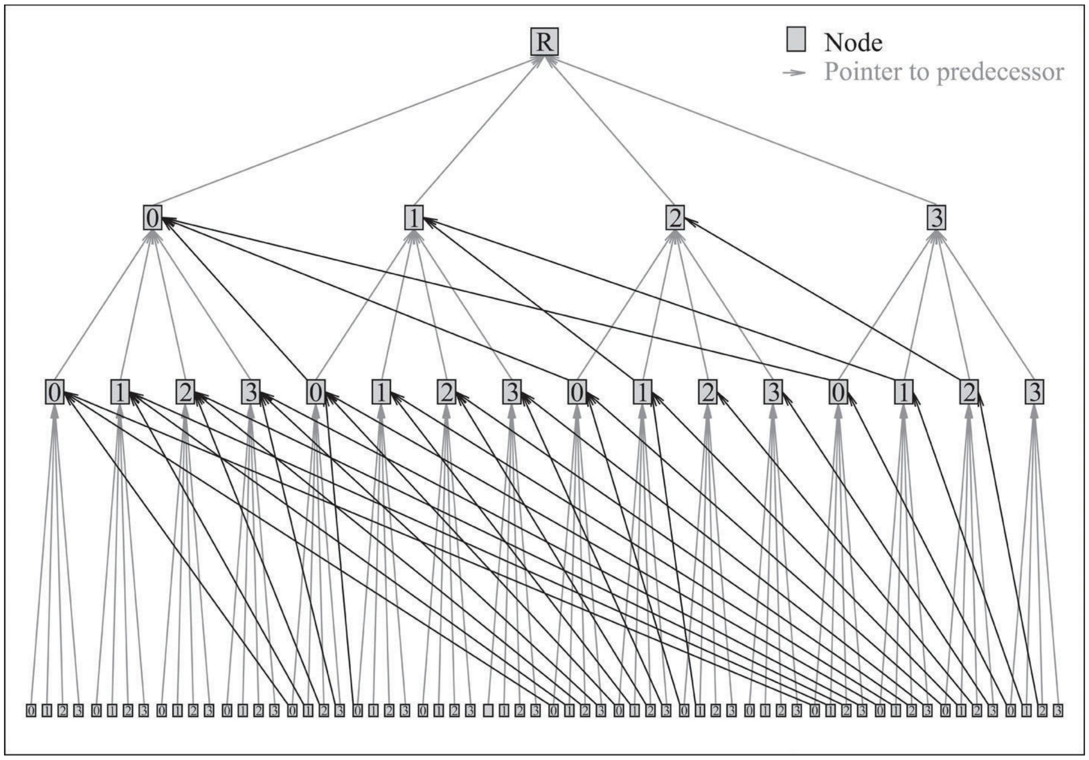

Figure 2.6: Fully populated keyword tree with left-pointing failure links.

Table 2.2: Production rules for the fully populated keyword tree (F).

Production rules
$F_0 \rightarrow 0$
$F_1 \rightarrow 1$
$F_2 \rightarrow 2$
$F_3 \rightarrow 3$
$F_4 \rightarrow F_0 \mid F_1 \mid F_2 \mid F_3$
$F_5 \rightarrow F_4\ F_4\ F_4$

II. Uniformly populated keyword tree (U).

Following the same steps, figure 2.7 shows the general structure of the uniformly populated keyword tree, while figures 2.8 and 2.9 show the right-pointing and left-pointing failure links for the same case, respectively.

Since this is a special case of the fully populated tree, it is no surprise to find the failure links following the same pattern as those of the previous case, i.e. pointing to the closest adjacent branch and backtracking one level only.

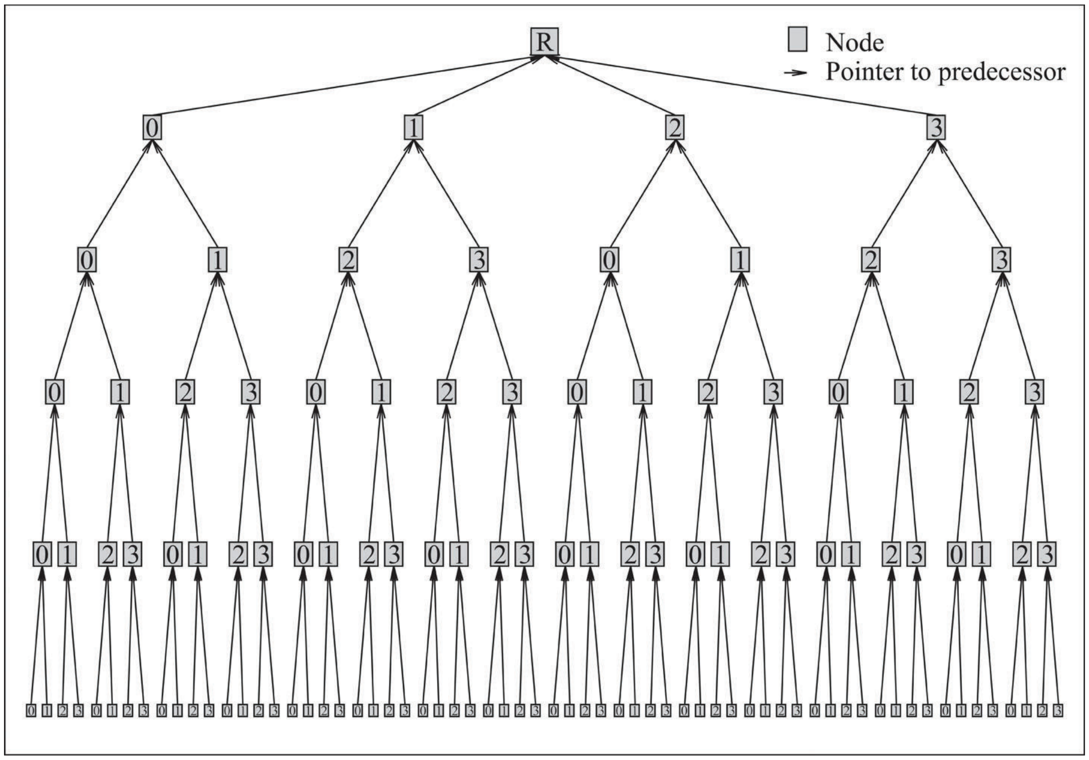

Figure 2.7: Uniformly populated keyword tree, height = 5.

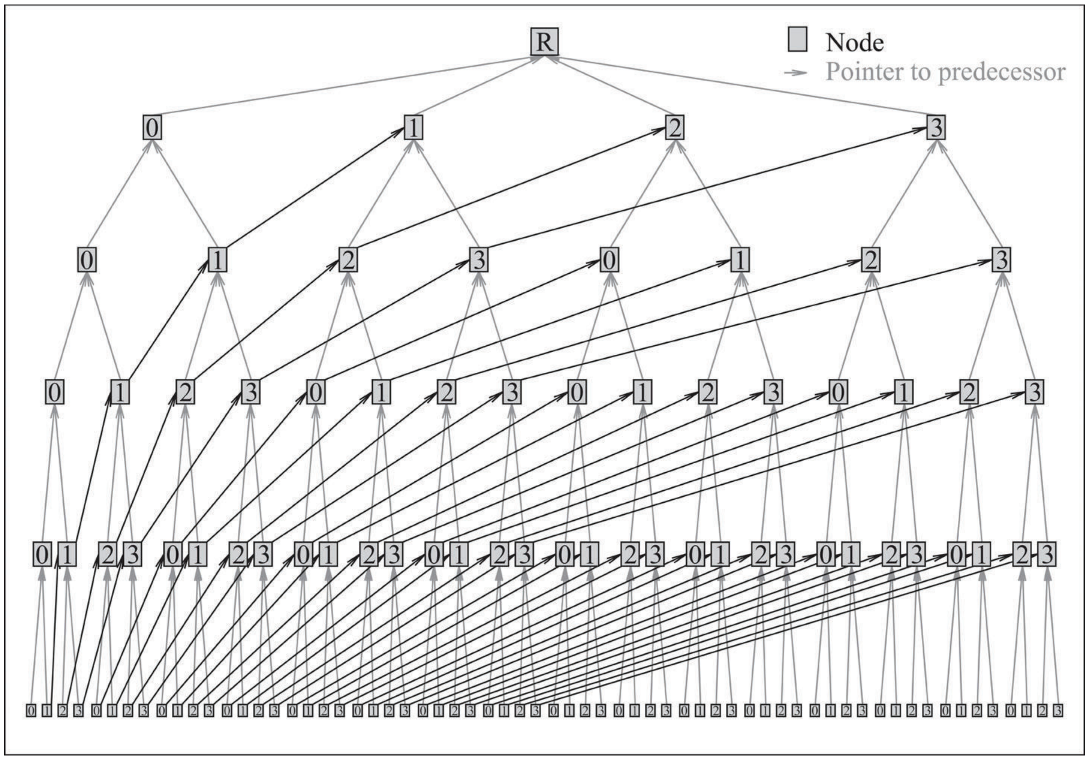

Figure 2.8: Uniformly populated keyword tree with right-pointing failure links.

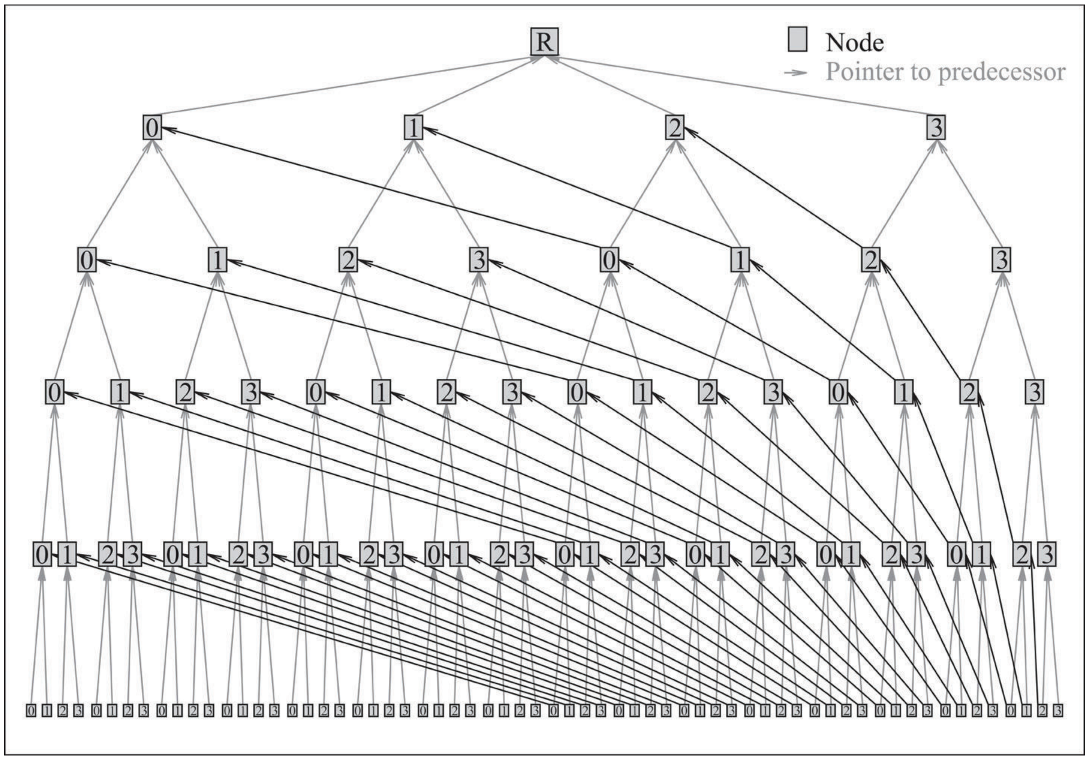

Figure 2.9: Uniformly populated keyword tree with left-pointing failure links.

Table 2.3: Production rules for the uniformly populated keyword tree (U).

Production rules	Production rules (cont.)
$U_0 \to 0$	$U_6 \to U_0\ U_4 \mid U_1\ U_5$
$U_1 \to 1$	$U_7 \to U_2\ U_4 \mid U_3\ U_5$
$U_2 \to 2$	$U_8 \to U_0\ U_6 \mid U_1\ U_7$
$U_3 \to 3$	$U_9 \to U_2\ U_6 \mid U_3\ U_7$
$U_4 \to U_0 \mid U_1$	$U_{10} \to U_0\ U_8 \mid U_1\ U_9$
$U_5 \to U_2 \mid U_3$	$U_{11} \to U_2\ U_8 \mid U_3\ U_9$
	$U_{12} \to U_0\ U_{10} \mid U_1\ U_{11}$
	$U_{13} \to U_2\ U_{10} \mid U_3\ U_{11}$
	$U_{14} \to U_{12} \mid U_{13}$

Table 2.3 shows the non-normalized production rules for the uniformly populated keyword tree (U).

III. Sparsely populated keyword tree (S).

Figure 2.10 shows the general structure of the sparsely populated keyword tree. As stated above, this is a cumulative version of the uniformly populated tree with entire sub-branches excluded. As a consequence, some of the failure links for this version traverse more than one tree level.

Figure 2.11 shows the single-level right-pointing failure links, figure 2.12 shows the single-level left-pointing failure links, figure 2.13 shows the multi-level right-pointing failure links, and figure 2.14 shows the multi-level left-pointing failure links for this keyword tree version.

Table 2.4 shows the non-normalized production rules for the sparsely populated keyword tree (S).

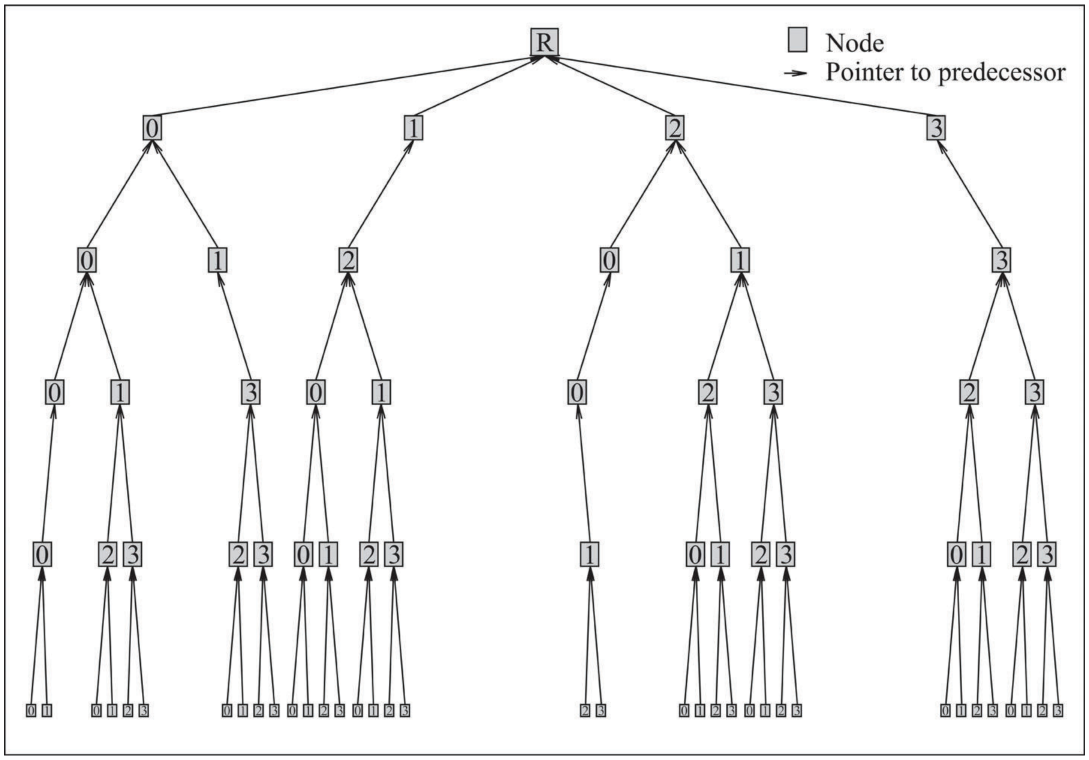

Figure 2.10: Sparsely populated keyword tree, height = 5.

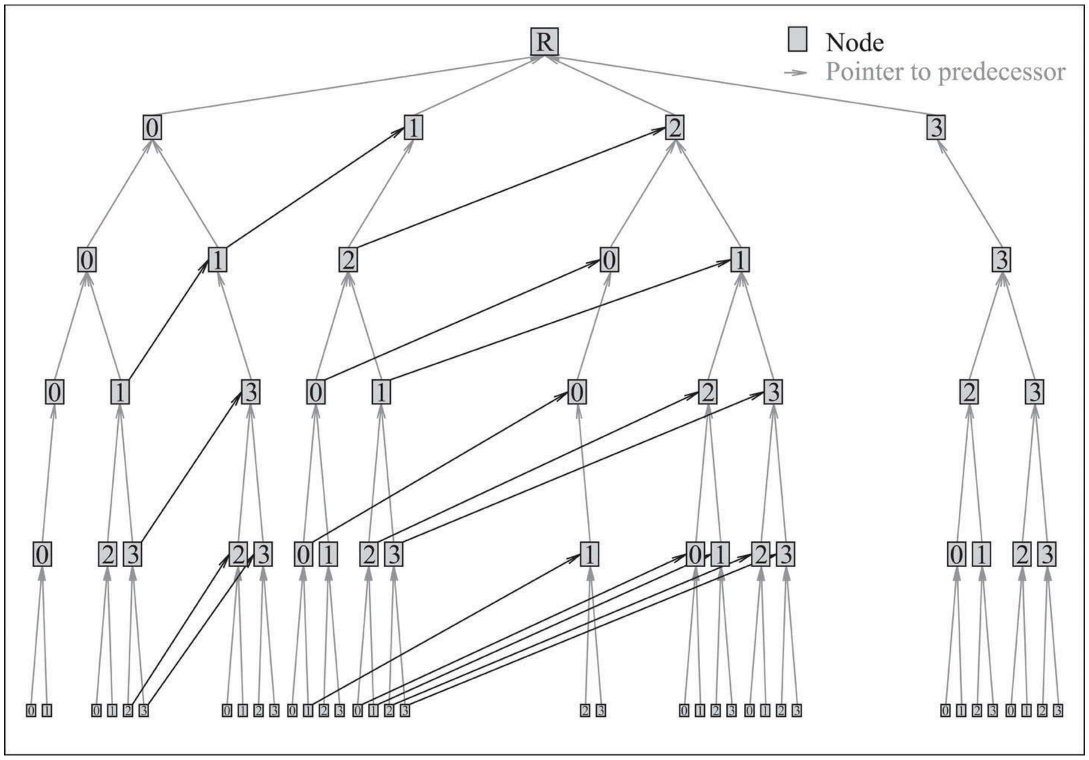

Figure 2.11: Sparsely populated keyword tree with single-level right-pointing failure links.

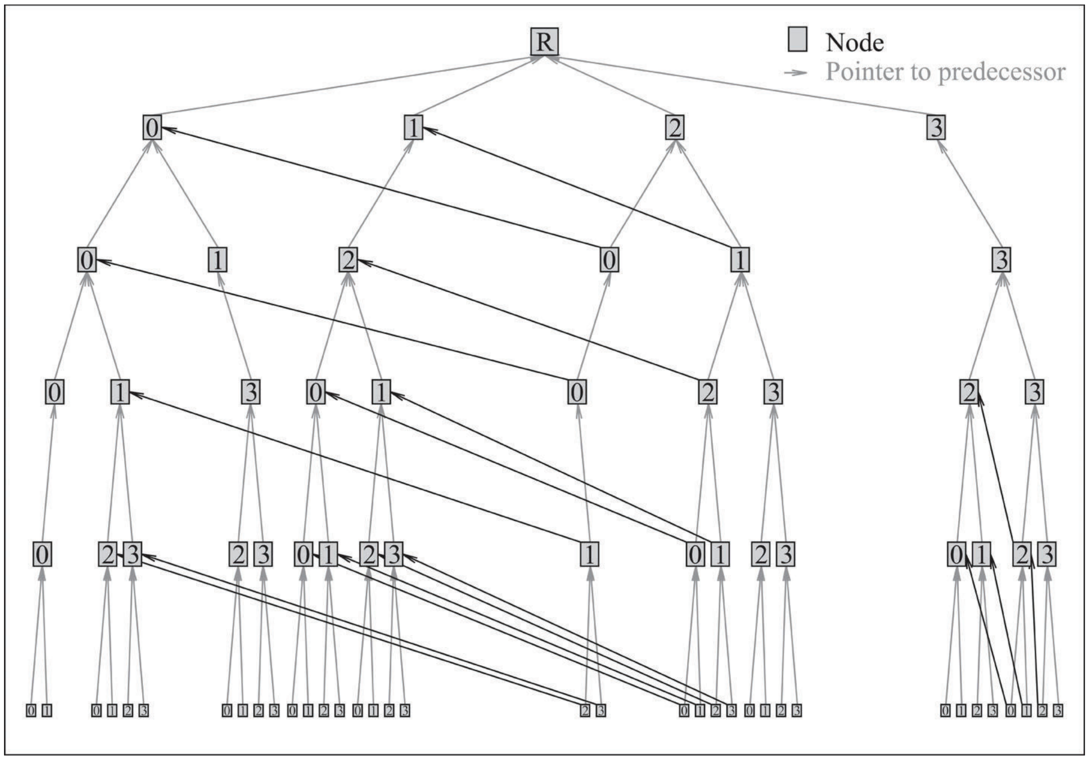

Figure 2.12: Sparsely populated keyword tree with single-level left-pointing failure links.

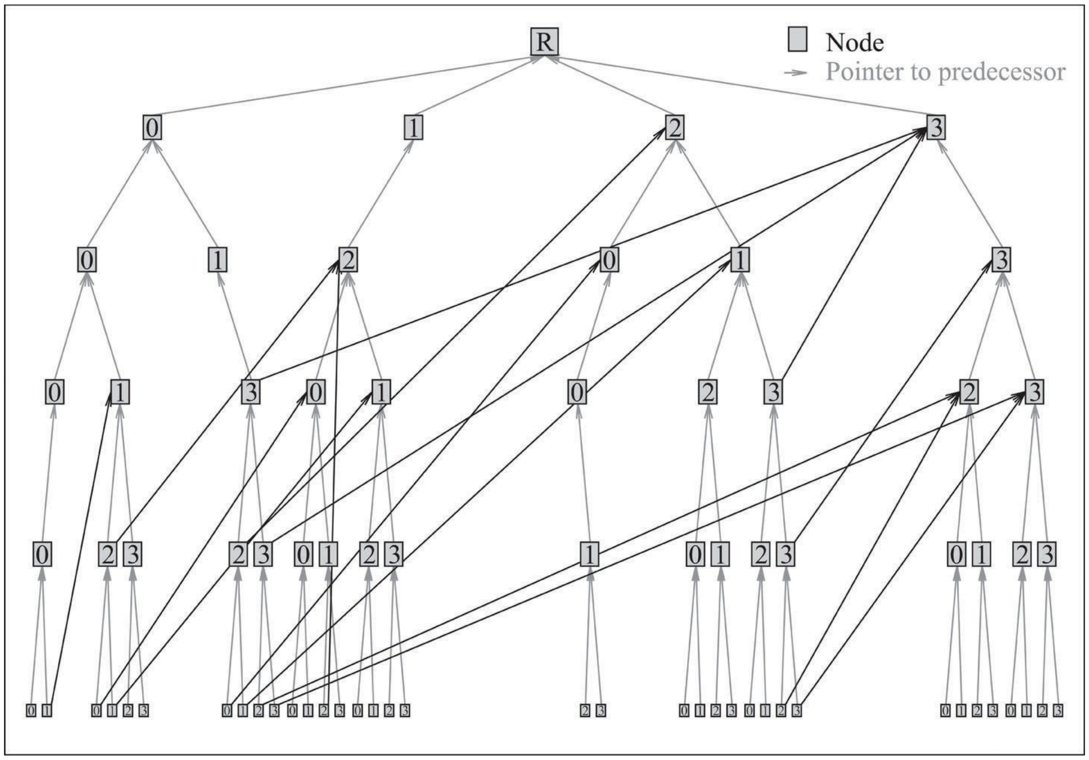

Figure 2.13: Sparsely populated keyword tree with multi-level right-pointing failure links.

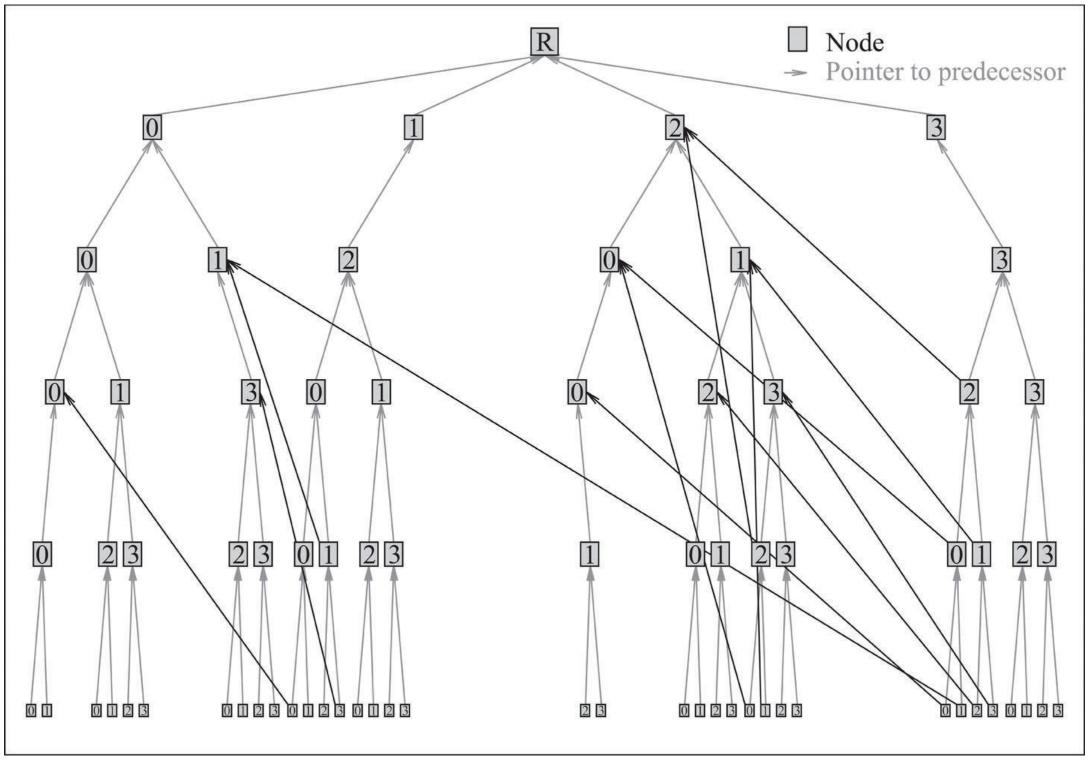

Figure 2.14: Sparsely populated keyword tree with multi-level left-pointing failure links.

Table 2.4: Production rules for the sparsely populated keyword tree (S).

Production rules	Production rules (cont.)
$S_0 \rightarrow 0$	$S_{10} \rightarrow S_0\ S_0\ S_0\ S_0\ S_4$
$S_1 \rightarrow 1$	$S_{11} \rightarrow S_0\ S_0\ S_1\ S_7$
$S_2 \rightarrow 2$	$S_{12} \rightarrow S_0\ S_1\ S_3\ S_7$
$S_3 \rightarrow 3$	$S_{13} \rightarrow S_1\ S_2\ S_8$
$S_4 \rightarrow S_0 \mid S_1$	$S_{14} \rightarrow S_2\ S_0\ S_0\ S_1\ S_5$
$S_5 \rightarrow S_2 \mid S_3$	$S_{15} \rightarrow S_2\ S_1\ S_9$
$S_6 \rightarrow S_0\ S_4 \mid S_1\ S_5$	$S_{16} \rightarrow S_3\ S_3\ S_9$
$S_7 \rightarrow S_2\ S_4 \mid S_3\ S_5$	$S_{17} \rightarrow S_{10} \mid S_{11} \mid S_{12} \mid S_{13} \mid S_{14} \mid S_{15} \mid S_{16}$
$S_8 \rightarrow S_0\ S_6 \mid S_1\ S_7$	
$S_9 \rightarrow S_2\ S_6 \mid S_3\ S_7$	

IV. Distinctively populated keyword tree (D).

Figure 2.15 shows the general structure of the distinctively populated keyword tree. Once more, since this is a special case of a fully populated keyword tree, and no specific sub-branches have being omitted, the failure links return to the pattern of pointing to the nearest adjacent branch and to backtrack the process by only one tree level as shown in figures 2.16 and 2.17, respectively.

Table 2.5 shows the non-normalized production rules for the distinctively populated keyword tree (D).

Table 2.5: Production rules for the distinctively populated keyword tree (D).

Production rules	Production rules (cont.)
$D_0 \rightarrow 0$	$D_{12} \rightarrow D_0\ D_{11} \mid D_1\ D_{10} \mid D_2\ D_9$
$D_1 \rightarrow 1$	$D_{13} \rightarrow D_0\ D_{11} \mid D_1\ D_{10} \mid D_3\ D_8$
$D_2 \rightarrow 2$	$D_{14} \rightarrow D_0\ D_{11} \mid D_2\ D_9 \mid D_3\ D_8$
$D_3 \rightarrow 3$	$D_{15} \rightarrow D_1\ D_{10} \mid D_2\ D_9 \mid D_3\ D_8$
$D_4 \rightarrow D_0 \mid D_1 \mid D_2$	$D_{16} \rightarrow D_0\ D_{15} \mid D_1\ D_{14} \mid D_2\ D_{13}$
$D_5 \rightarrow D_0 \mid D_1 \mid D_3$	$D_{17} \rightarrow D_0\ D_{15} \mid D_1\ D_{14} \mid D_3\ D_{12}$
$D_6 \rightarrow D_0 \mid D_2 \mid D_3$	$D_{18} \rightarrow D_0\ D_{15} \mid D_2\ D_{13} \mid D_3\ D_{12}$
$D_7 \rightarrow D_1 \mid D_2 \mid D_3$	$D_{19} \rightarrow D_1\ D_{14} \mid D_2\ D_{13} \mid D_3\ D_{12}$
$D_8 \rightarrow D_0\ D_7 \mid D_1\ D_6 \mid D_2\ D_5$	$D_{20} \rightarrow D_0\ D_{19} \mid D_1\ D_{18} \mid D_2\ D_{17}$
$D_9 \rightarrow D_0\ D_7 \mid D_1\ D_6 \mid D_3\ D_4$	$D_{21} \rightarrow D_0\ D_{19} \mid D_1\ D_{18} \mid D_3\ D_{16}$
$D_{10} \rightarrow D_0\ D_7 \mid D_2\ D_5 \mid D_3\ D_4$	$D_{22} \rightarrow D_0\ D_{19} \mid D_2\ D_{17} \mid D_3\ D_{16}$
$D_{11} \rightarrow D_1\ D_6 \mid D_2\ D_5 \mid D_3\ D_4$	$D_{23} \rightarrow D_1\ D_{18} \mid D_2\ D_{17} \mid D_3\ D_{16}$
	$D_{24} \rightarrow D_{20} \mid D_{21} \mid D_{22} \mid D_{23}$

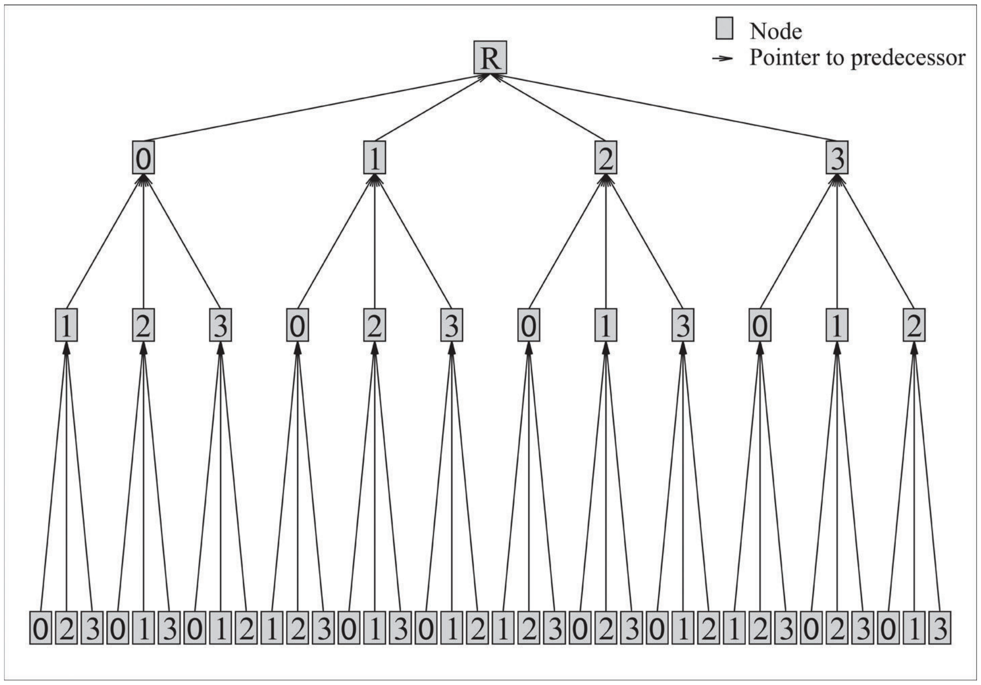

Figure 2.15: Distinctively populated keyword tree, height = 3.

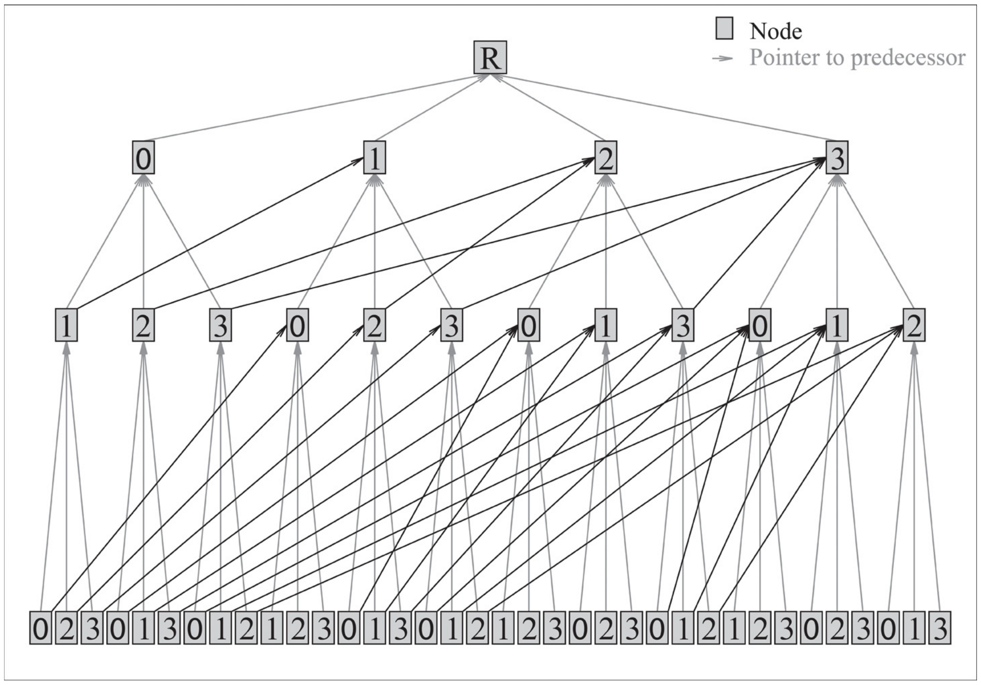

Figure 2.16: Distinctively populated keyword tree with right-pointing failure links.

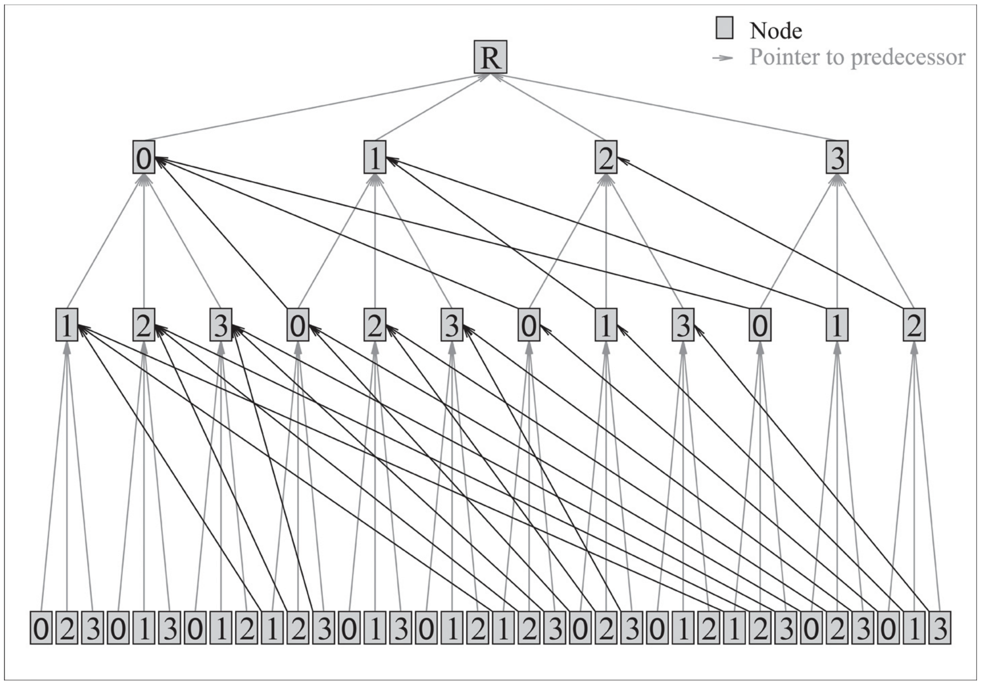

Figure 2.17: Distinctively populated keyword tree with left-pointing failure links.

V. Generalizing.

Consider for a moment the case in which all keyword trees should accept words of the same length (i.e. subsequences composed of the same quantity of individual characters or symbols). From the computer sciences perspective, this is the equivalent of having search trees in which all paths from the root to any leaf have the same height.

Notice that all production rules for the trees presented so far have the same height, except for the fully populated tree. To make it the same height as all the rest, simply witch rule F_5 in table 2.2 for the following new rule: $F_5 \rightarrow F_4\ F_4\ F_4\ F_4\ F_4$

From a side-by-side comparison of the production rules necessary to generate each one of the keyword trees described above, it can be observed that, as the specialization of the tree increases, the number of production rules required to specify their respective regular grammar increases as well. This fact agrees with the notion that, each new version of the keyword tree presented thus far requires a higher level of control on what terminal symbol could follow a previously selected terminal symbol. Therefore, as the selectivity of the tree increases, it is expected that the regular grammar to describe it should increase in complexity. This is equivalent to saying that while the set of individual words accepted by the grammar decreases (i.e the dictionary set decreases), the number of production rules to represent this higher level of specialization increases.

At some point, this increased complexity could reach a degree where it is no longer practical or efficient to use a set of production rules to analyze any given text, in order to find the words from a dictionary set. Then, in practical implementation terms, it will be much easier to construct the corresponding tree directly from the dictionary set, and transform the identification task into a matching problem between the text of interest and the newly created keyword tree. This will be done by taking one character from the text at a time, and traversing the keyword tree for the corresponding match or mismatch (as stated at the beginning of this subsection).

2.5 Analysis theories of moving objects.

After careful consideration of all the bibliographic references included in appendix B, it was decided that the theoretical frameworks for analysis of moving object patterns are still limited to special cases. In consequence, none of them were used in the present work (directly or indirectly). Additional details of the advancement achievements of each framework, along with limitations identified by their creators, appear in appendix B.

2.6 Chapter summary.

This chapter has presented past studies whose conclusions are considered relevant to the objectives of the current work.

First, environmental studies on grizzly bears were examined to identify some of the factors that influence their habitat choices and some of their behaviors (section 2.1). Particular attention was given to technical considerations that have improved the objectivity of the conclusions of this type of studies such as the use of GPS collars for tracking purposes and satellite imagery to generate digital maps of the study areas (e.g. land cover maps, digital elevation maps). Some of the studies contributed theoretical and empirical results to compensate for over and under representation of certain location types (e.g. shrub vs. high density canopy, respectively), while some others considered mathematical models to artificially supplement the acquired movement data with what can be considered interpolation methods.

Section 2.2 identified some of the major historical changes on relational database technologies, to propose contemporary RDBMS systems as robust and mature solution platforms. Some recent studies on implementation of complex indexing methods and similarity searches in very large databases, e.g. [209, 210], have concluded that external function libraries are an excellent option when complex data structures have to be implemented, because despite their efficiencies, RDBMSs offer limited programming alternatives for fully integrated equivalents. These studies identify the performance gains derived from full integration, i.e. pinning, buffering, etc., as unaccessible to the external libraries. This research believes that those same performance enhancement techniques are a potential advantage for a fully integrated implementation.

As a first approach, methods coming from the LBS specialty were presented as candidate solutions, based solely in their ability to detect frequent common subsequences in large collections of longer trajectories (see appendix A).

Then, the general conceptual structure of the KDD methodology was presented in section 2.3 as the preferred method for this work, based mainly on its flexibility as an iterative process. After the consideration of a simulated annealing solution conforming to this framework (see section A.2), section 2.4 identified the local alignment approach as a generally suitable method for the objectives of the current work, embedded as part of the KDD process.

Additional performance gains can be attained via carefully analyzed and implemented data structures, supporting the notion that whatever the method selected to process large and complex datasets, it has to be flexible and comprehensive. Flexibility has to be passed on to the user in terms of control parameters to allow the researchers to include or exclude any number of predefined patterns in their analysis, leaving the method to create a comprehensive catalog of the rest of the patterns present in the data that the researchers did not expressly include or exclude.

Most of the conclusions of the computer science studies in this chapter can be summarized in the following common points:

- The larger the amount of individual factors included in the studies, the longer it will take to process the entire dataset with increasingly more complex methods.

- The larger the dataset, the larger the efficiency gains from careful design of the data structures to store and process the data.

This chapter included studies on grizzly bear environment and behavior (section 2.1), to identify the dominant factors that influence the preferences of the species, in order to answer the first objective of this work. The rest of this chapter reviewed technical specifications and computer sciences' studies that helped in refining the requirements, constraints and expected features of the automated solution generally defined by the objectives of this work.

Finally, mainly because of their limitations as solutions for special cases, analytical moving object frameworks were moved from section 2.5 to appendix B, and were no longer considered in any other form in the implementation of this work. The included publications in appendix B helped to answer the sixth objective of this work (see section 5.1 for further details).

Chapter 3

Methods

With the exception of sections 3.1, 3.2, and 3.7.1, the concepts presented in this chapter follow the KDD methodology proposed by Han and Kamber [128] (detailed in section 2.3). Its objective is to determine all technical aspects of the necessary tasks to analyze the grizzly bear tracking dataset with machine learning methods.

All methods in this work were implemented using Microsoft SQL Server 2005 Express. The DBMS is fully featured, but limited to a total database size of 4 GB.

All DBMS programming was done in Transact-SQL (T-SQL), and all data, including temporary data structures for all methods, were stored in relational table form.

As stated in section 2.3, data mining is an iterative method that requires repeated experimentation not only to calibrate the software tools, but also to achieve convergence depending on the needs of the individual approaches. As such, some of the tasks in the different subsections were added as part of the implementation of the various machine learning methods, specially in the segmentations' section 3.6.1. However, because of the hierarchical organization of the KDD process by Han and Kamber [128], all of those additions are presented in their corresponding section, and not in chronological order.

3.1 The overall approach.

Once the unsuitability of the location-based approach (subsection A.1.1) and simulated annealing (subsection A.2.1) methods were established, this work focused on the implementation of the method by Altschul et al. (see subsection 2.4.1.2).

This method requires an initial set of small subsequences to start the search against the database. In Biotechnology, these usually correspond to amino acid groups of known physical or chemical properties. In the case of grizzly bears, no equivalent movement subsequences are known, instead the following analysis steps were taken for the five variations of the daily segmentations (subsection 3.6.1.4) combined with two versions of the log-likelihood ranking matrix (asymmetric and symmetric, subsection 2.4.1.2) to determine subsequences of interest.

- Take up to seven random samples per segment.
- Retain only unique subsequences (discard all but one repetition of the same subsequence).
- Rank the unique subsequences with the log-likelihood matrix.
- Choose the upper quartile (by log-likelihood rank) as the initial subsequence set.
- Create the Reversed Keyword Tree (RKT) for the set (see subsection 3.7.2.3).
- Create the corresponding mismatch matrix.
- Find all matches (including overlapping).
- Find the segments with multiple non-overlapping matches (clustering).

To determine sensitivity of the method, nine subsequence sizes (3, 6, 9, 12, 15, 20, 25, 30 and 40 symbols) were processed by repeating all the steps in the previous list.

3.2 Study area and tracking dataset.

The dataset used for this work was provided by the Foothills Research Institute [107] and includes tracking data of grizzly bears in the province of Alberta, Canada. Specifically, a subset of tracking collar positions (GPS collar position fixes) corresponding to the Foothills region of the Canadian Rocky Mountains. The data was acquired between the years of 1999 and 2008 including biological indicators from direct examinations of the tracked individuals (e.g. age, gender, reproductive status, weight, size measurements, blood samples).

Figure 3.1 highlights the geographical location of the province of Alberta in relationship to the rest of the Canadian provinces and territories, as well as the 48 contiguous states of the USA. In order to show the entire extension of Canada, the Universal Transverse Mercator N15 (UTM N15) projection was used to generate this map. Due to the projection's limitations, the outline of the USA's state of Alaska is missing from this representation (to the Northwest of the Yukon territory). The rest of the maps in this thesis were generated using the UTM N11 projection.

The collar models changed from year to year and included instruments that were programmed to acquire GPS fixes every one, two or four hours, and with most newer systems collecting data every 20 minutes. Throughout these years, some collars were programmed to acquire position fixes at irregular intervals.

Spatial datasets included digital maps of the same geographic region obtained from either previous studies on the same subject (e.g. categorized land cover raster at 30m of spatial resolution by Franklin et al. [110]) or from the Canadian government (e.g. shapefile of major towns). Figure 3.2.a shows the extension of the study area in the context of the province including its major towns, rivers and roads, while figure 3.2.b shows the natural regions in the province as defined by the Natural Regions Committee [184].

3.2.1 Physical description.

The study area has a total extent of 259,964 km^2 including five of the six natural regions defined by the Natural Regions Committee of Alberta [184]. The "Canadian Shield" is the only natural region not included in the study area.

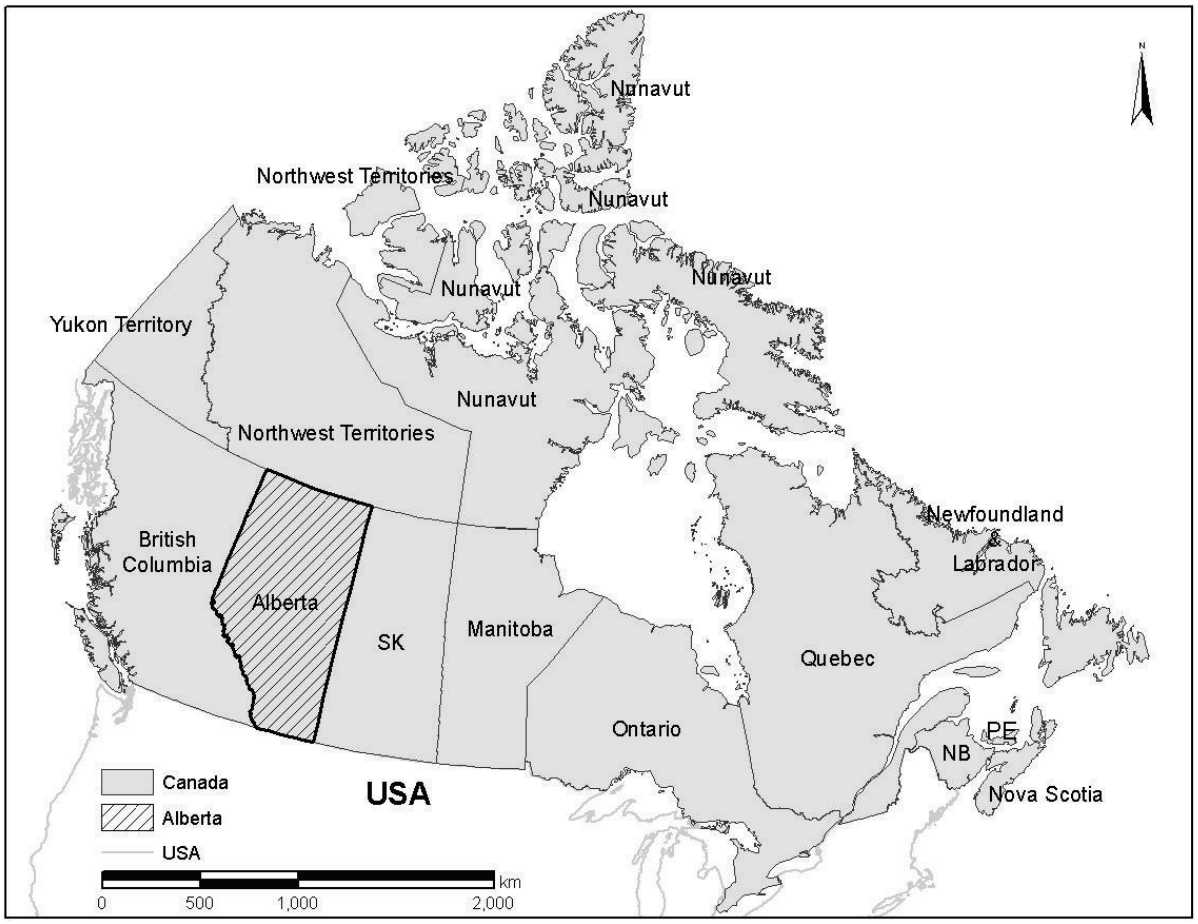

Figure 3.1: Geographical location of the province of Alberta, Canada.

Table 3.1 includes the natural regions and subregions as defined by the Natural Regions Committee [184] and their equivalent ecoregion in the earlier classification by Strong [233], showing that the newer subregions are environmental classification refinements from the earlier studies.

According to the Natural Regions Committee [184] each of the natural regions in the study area is characterized in the following way.

Boreal Forest: "Aspen forests with shrubby understories, coniferous forests, white spruce, jack pine on dry and sandy sites, closed-canopy mixedwood, and dry jack pine forests. Dune areas are largely unvegetated. Black spruce (tamarack stands), balsam poplar, paper birch, lodgepole pine, jack pine hybrids. Peatlands, graminoid marshes and

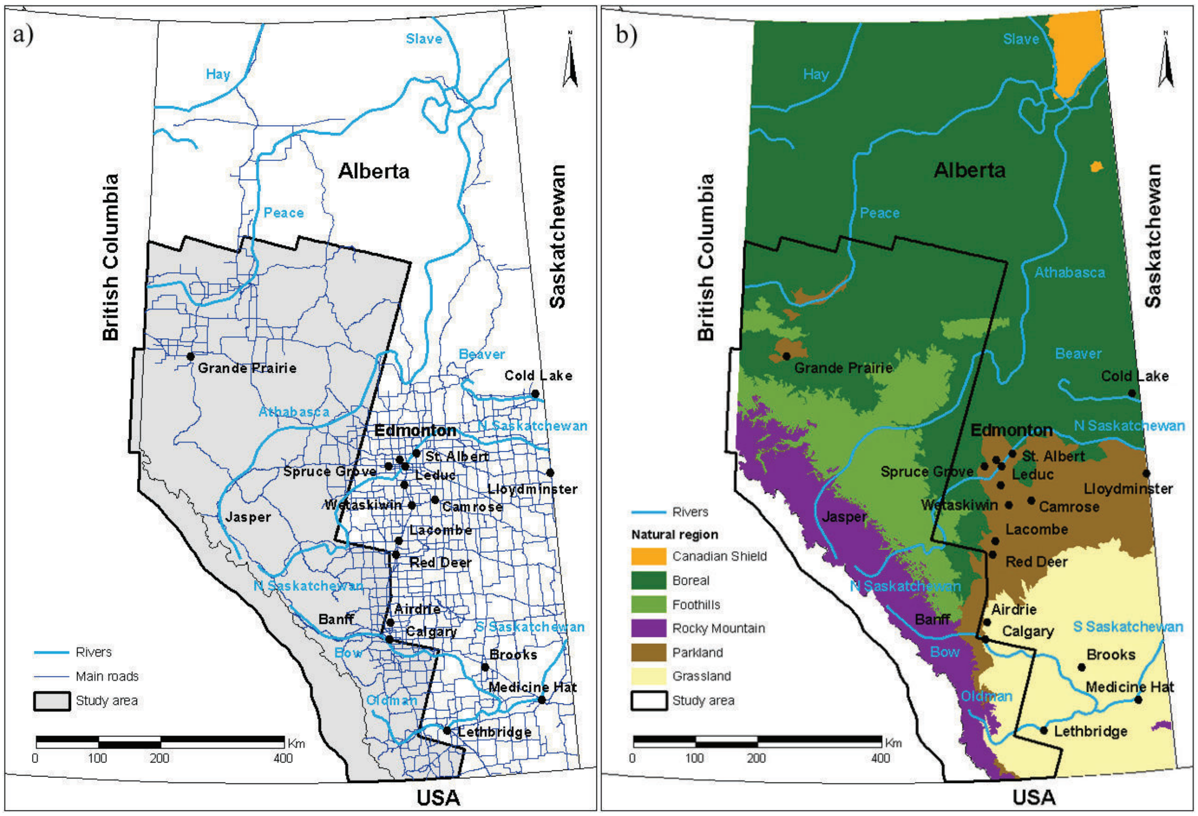

Figure 3.2: Province of Alberta, Canada, with a) main towns, roads, rivers and outline of the study area, and b) natural regions (based on [184]).

willow/marsh reed grass wetlands, sedge meadows and sedge fens. Aquatic, shoreline, meadow, shrub and marsh vegetation in the lowlands. Upland forests and shrubs on terraces, islands and levees." [184]

Foothills: "Mixedwood forests and closed coniferous forests. Aspen, lodgepole pine, white spruce and black spruce." [184]

Rocky Mountain: "Pure aspen forests, mixed aspen forests, mixed conifer forests, herbaceous meadows and shrublands. Aspen, lodgepole pine, Douglas fir, white spruce, and Engelmann spruce. Grasslands and largely non-vegetated areas." [184]

Parkland: "Aspen clones interspersed with grasslands dominated by plains rough fescue and remnant aspen clones and continuous forest. Aspen forests (continuous and clones), some areas of dense tall willow (north), California oatgrass, porcupine grass and

Table 3.1: Natural regions, subregions and ecoregions of the study area.

Natural region	**Natural subregion**	**Ecoregion**
Canadian Shield	Kazan Uplands	High Boreal Mixedwood
Boreal Forest	Athabasca Plain	
	Boreal Subarctic	Boreal Subarctic
	Central Mixedwood	Mid Boreal Mixedwood
	Dry Mixedwood	Low Boreal Mixedwood
	Lower Boreal Highlands	
	Northern Mixedwood	
	Peace-Athabasca Delta	
	Upper Boreal Highlands	
Foothills	Lower Foothills	Lower Boreal-Cordilleran
	Upper Foothills	Upper Boreal-Cordilleran
Rocky Mountain	Montane	Montane
	Subalpine	Subalpine
	Alpine	Alpine
Parkland	Central Parkland	Aspen Parkland
	Foothills Parkland	
	Peace River Parkland	
Grassland	Dry Mixedgrass	Dry Mixed Grass
	Foothills Fescue	Fescue Grass
	Mixedgrass	Mixed Grass
	Northern Fescue	

jack pine on sands. Extensively cultivated, graminoid wetlands often ringed by willow. Grasslands more common on southerly slopes.” [184]

Grassland: “Blue grama, needle, thread, porcupine grass, western porcupine grass (drier), northern and western wheatgrass (on drier sites). Shrublands in moister locales, plains rough fescue (moist), and mountain rough fescue on moister sites. Mainly agricultural, buckbrush, rose shrublands, graminoid wetlands and wet areas often shrubby.” [184]

3.3 Data cleaning.

After a first analysis of the GPS positions’ database, it was established that only some bears were tracked during several years. Figure 3.3 shows the multi-year tracking data

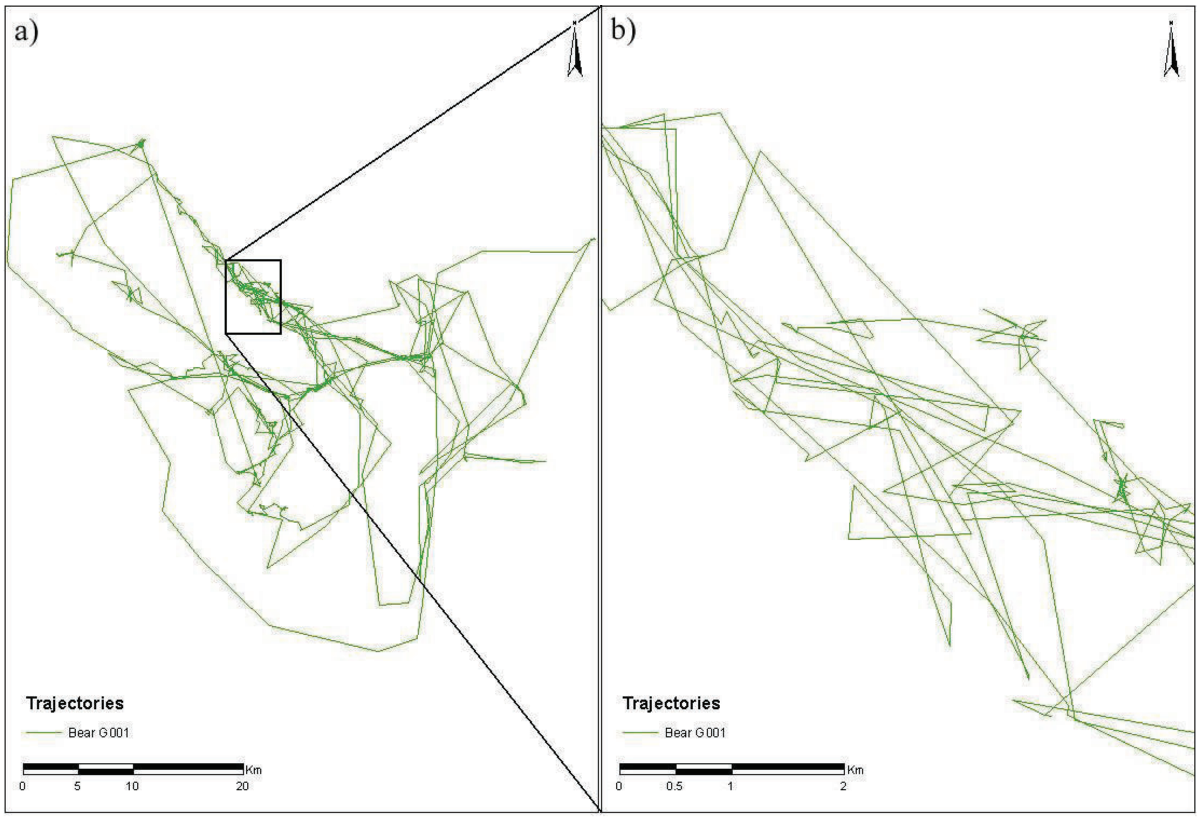

Figure 3.3: Trajectories for bear G001 a) Multi-year, b) Detail.

for bear G001 of the study. The individual consecutive positions have been connected by a straight line for an initial assessment of the complexity of the trajectories and the distance between points.

This data was provided in database form with a single table containing all the details of each GPS position.

This analysis identified the following list of major inconsistencies in the data.

- Invalid timestamp (e.g. no timestamp at all, nonexistent date or time).
- Annual trajectories with more than one point with the same timestamp.
- Annual trajectories with 10 or less points per year.
- Temporal discontinuities between consecutive points of more than five times the

mean temporal resolution of the annual trajectory*.

For the first two problems in the list, the erroneous individual positions were excluded from the trajectories and the analysis continued.

The data that presented the last two inconsistencies in the previous list constituted 1.05% of the uncleaned dataset, making it a comparativelly small portion of the data, therefore defining special cases. The inclusion of those special cases in the automated solution would have required special sections of executable code for their analysis, in addition to the necessary code for the analysis of the majority of the data.

Since one of the main objectives of the analysis included identifying frequent movement patterns, in contrast to uncommon ones, this work judged their temporary exclusion as a measure to arrive at a general analysis tool in a shorter amount of time.

To detect the fourth problem in the list, annual trajectories for each bear were analyzed individually. The mean time interval between consecutive positions of each annual trajectory (a.k.a. the mean temporal resolution of the annual trajectory) was calculated independently, and that value was compared against the time interval between every pair of consecutive GPS positions. Every annual trajectory with at least one pair of points with a time interval greater than five times the mean temporal resolution of the annual trajectory was flagged for further analysis.

For the flagged annual trajectories, two more metrics were calculated. The percentage of the points with discontinuities against the total number of points in the trajectory, and the percentage of time contributed by the points with discontinuities against the total time duration of the trajectory. If the percentage of points with discontinuities was 1% or less, and if those points contributed 20% or more of the total trajectory time, that trajectory was excluded from the cleaned dataset, otherwise the trajectory was manually analized for a final decision.

*E.g. more than 20 hours for collars programmed to acquire a GPS fix every four hours.

39 annual trajectories presented the forth problem of the list. Of those, 3 were automatically excluded from the cleaned dataset by the combined rule of the previous paragraphs. The remaining 36 trajectories were analyzed manually, and 32 of them were still accepted in the cleaned dataset because there was no clear indication of systematic errors in them. Summarizing, only 7 annual trajectories in total were excluded from the cleaned dataset because of the fourth inconsistency of the list.

3.3.1 Conceptual temporal resolution of the cleaned dataset.

Analyzing the distribution of time intervals between consecutive positions in the entire cleaned dataset, as shown in figure 3.4, a few observations are in order.

- There are relatively minor accumulations of points, of a few thousand points per measure, for intervals of 3 and for every interval between 5 and 16 hours.
- Except for the 3 hours interval, these minor accumulations decrease in frequency as the time interval increases.
- An intermediate accumulation of 8,835 points, occurs at the 2 hours interval.
- Major accumulations of 17,355 and 33,654, occur at 4 and 1 hour intervals, respectively.
- The largest accumulation of points, 43,608, occurs at the 20 minutes interval.

This distribution is generally consistent with the GPS collar model progression and position acquisition programming described in section 3.2.

Based on these observations, it can be concluded that the highest temporal resolution data in the dataset was captured with intervals of 20 minutes between consecutive position fixes. Which can be used as the conceptual temporal resolution of the cleaned

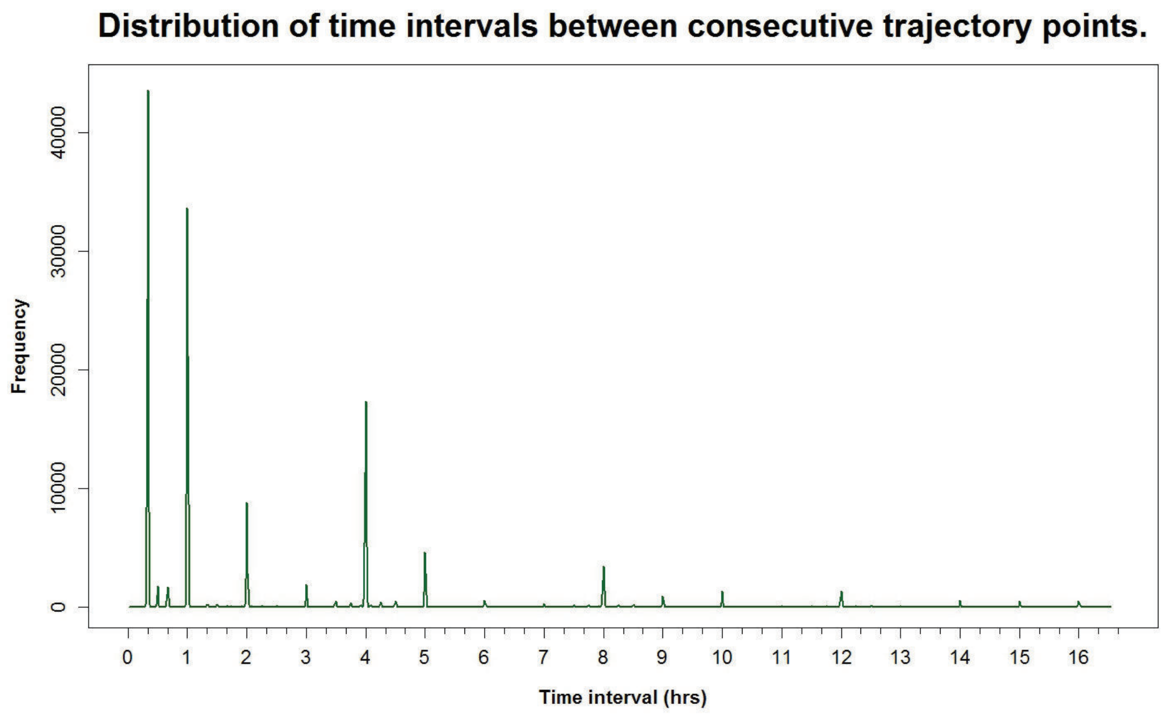

Figure 3.4: Frequency distribution of time intervals between consecutive trajectory position fixes.

dataset for the purposes of subsection 3.6.1. It also has the added advantage of being an integer sub multiple of all the other time intervals at which high concentrations of consecutive positions were acquired.

3.4 Data integration.

Simple geographical intersections of the GPS positions and the digital maps (one at a time) were calculated using ESRI ArcCatalog 9.3 and ArcMap 9.3 (“n/a” $\equiv 0$, for no intersection). The resulting data was stored in the original GPS position table adding one column to store the matched supplementary data (e.g. the elevation at which every GPS position was acquired). Figure 3.5 shows a Digital Elevation Model (DEM) and a categorized land cover map matched to the geographical extent of the study area as

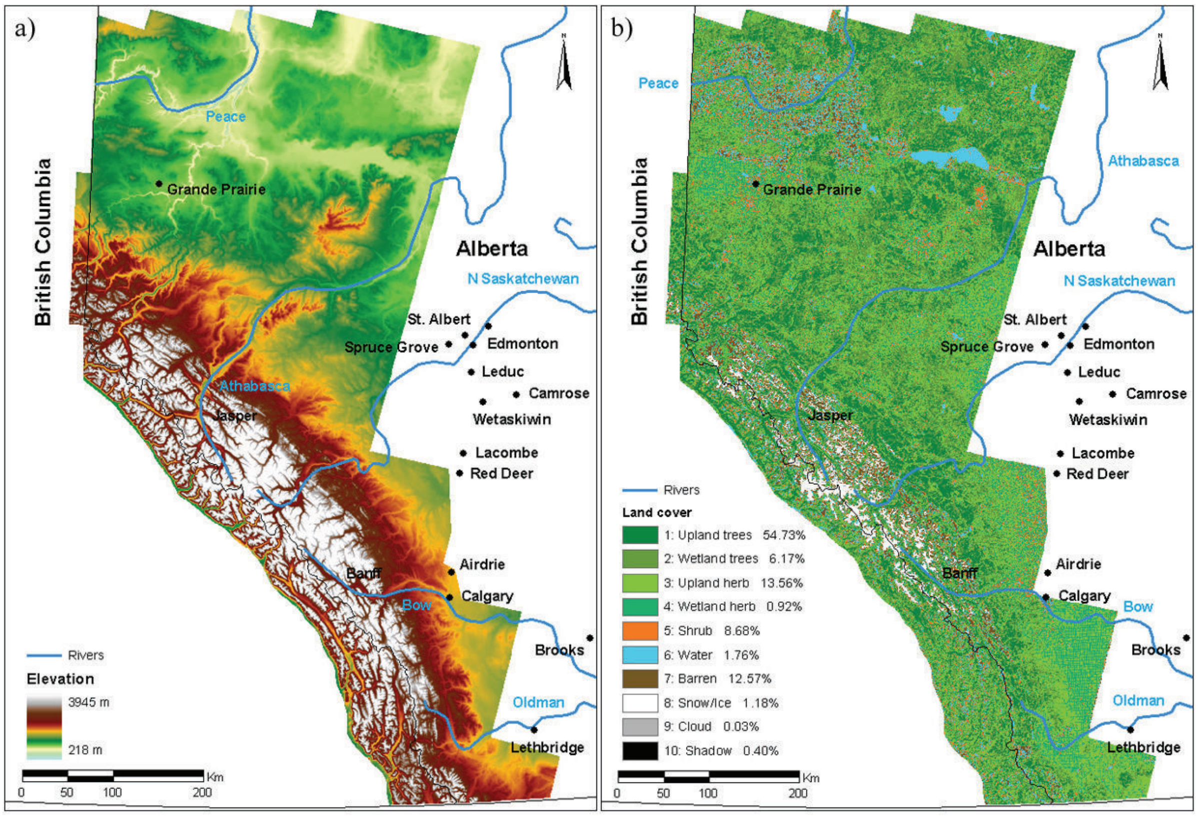

Figure 3.5: Study area, a) Elevation, b) Land cover categories.

closely as possible. The land cover categories were specifically generated by Franklin et al. [110] to reflect the terrain usage of grizzly bears.

Please note the legend in figure 3.5.b that shows the total area occupied by each land cover class in the full extent of the study area. Similar totals can be obtained by matching the GPS positions with the land cover map to establish relative abundance of each land cover class, table 3.2 shows both relative percentages. From the side-by-side comparison of these percentages it can be observed that the "Shrub" land cover is over-represented in the GPS positions in relation to its abundance in the full study area. In contrast, the "Water" class is under-represented in the GPS data by a factor of approximately 11.

Using the percentages in table 3.2, combined with the total size of the cleaned and integrated dataset of 199,223 individual positions, returned a χ^2 [78, 108] value of 57,829.05*.

*See appendix C.2 for the tabular results of this test.

Table 3.2: Relative abundance of land cover classes.

Symbol	Land cover	Full study area (%)	GPS positions (%)
1	Upland trees	54.7318	53.2899
2	Wetland trees	6.1725	1.8844
3	Upland herb	13.5581	10.9417
4	Wetland herb	0.9159	0.5407
5	Shrub	8.6798	22.5710
6	Water	1.7615	0.1639
7	Barren	12.5657	10.3703
8	Snow / Ice	1.1801	0.0822
9	Cloud	0.0314	0.0030
A	Shadow	0.4032	0.1528
	Total	100.0000	100.0000

Including the "n/a" class (not shown in table 3.2), comparing this value with the tabulated χ^2 values for ten degrees of freedom (29.59 for $p = 0.001$), it was observed that over and under-representation of classes was to be fully expected in this analysis. In other words, that the observed GPS-land cover matched position's distribution did not agree with the study area's land cover distribution.

The machine learning methods used to analyze this dataset had to offer options to compensate for such conditions.

All other supplementary data provided in tabular form (spreadsheet or database table), e.g. gender and reproductive status data, was integrated by database inner joins.

3.5 Data selection.

Detailed analysis on the temporal dimension of the GPS positions revealed the near absence of data for the months of January, February, March and December of every year, which coincides with the denning period of the species, when all individuals remain static [40, 46, 58, 115, 183, 190]. Figure 3.6 shows a daily count of GPS points from all bears, while figure 3.7 shows a monthly accumulation. Both figures also show a maximum density of GPS points between the months of June and September of every year (inclusive), which coincide with the period of the year when most grizzly bear food sources are at maximum production [40, 46, 58, 115, 183, 190].

The dataset was divided in two parts, a calibration dataset for software implementation and testing purposes, and the rest of the data for validation.

As far as this research is aware, there is no single recommendation about the best size for the calibration dataset on research of this nature, but some projects with geographically referenced data select as little as 5% of their dataset for calibration tasks (e.g. [44, 45]). The following subjective method was followed to select the calibration dataset with two simultaneous objectives, make the calibration dataset as close as possible to 20% of the entire dataset to make software development as fast as possible, and include more than one trajectory of each "size class" per year.

As described in section 3.3, two metrics were calculated for each annual bear trajectory, the total number of individual GPS points, and the mean time interval between consecutive position fixes (i.e. the mean temporal resolution of the annual trajectory).

For each year, all trajectories were sorted by their total number of positions, and the maximum, minimum and yearly mean number of positions were calculated. Using these values, three "size classes" were roughly defined as, "trajectories with the most number of positions" (large), "trajectories with the least number of positions" (small), and "trajectories with the mean number of positions" (mean), without any formal transition value between them.

The calibration dataset was chosen to include entire trajectories from each year with iterative algorithm 3.1.

The average temporal resolution of the annual trajectories was used in the search processes to avoid, as much as possible, the inclusion of multiple trajectories for the same year with the same temporal resolution. This additional selectivity was imple-

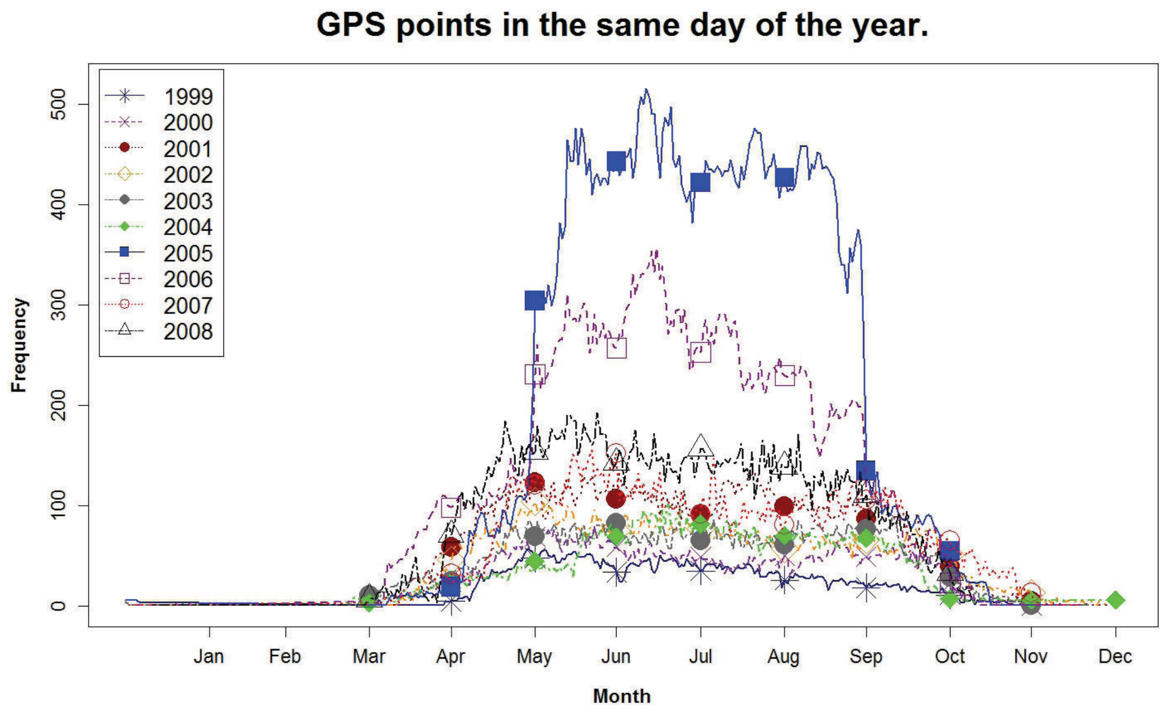

Figure 3.6: GPS points by day of the year.

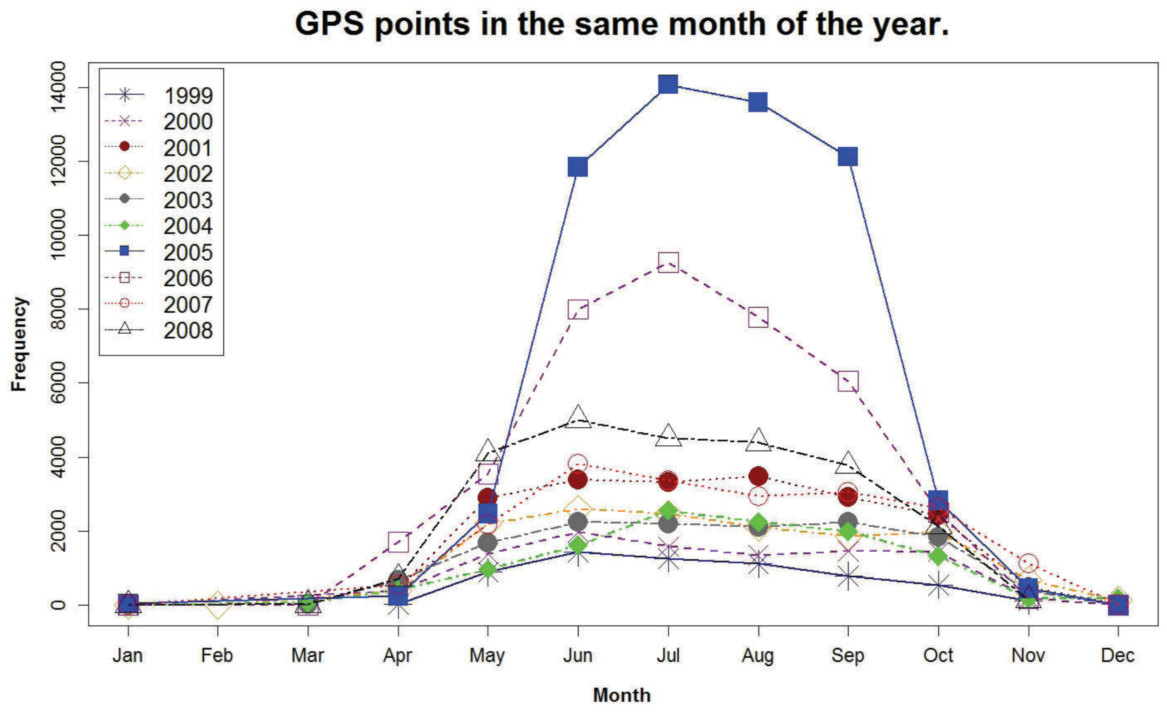

Figure 3.7: GPS points by month of the year.

Algorithm 3.1 Calibration dataset selection

```
Input: The full dataset of trajectories, desired_size
Output: The list of calibration trajectories or an error code.
 1: Mark all trajectories as unexamined.
 2: $Current\_Size \leftarrow 0$
 3: repeat
 4:     Choose one of the years of the full dataset (in circular fashion).
 5:
 6:     Find the next unexamined large trajectory.                 ▷ Downwards search
 7:     if Found then
 8:         Mark the trajectory as included.
 9:         $Current\_Size \leftarrow$ current size of the calibration dataset.
10:         if $Current\_Size = (desired\_size \pm 2\%)$ then
11:             Break.
12:         else if $Current\_Size > (desired\_size + 2\%)$ then
13:             Mark the trajectory as excluded.
14:         end if
15:     end if
16:
17:     Find the next unexamined mean trajectory.                  ▷ Downwards search
18:     if Found then
19:         Mark the trajectory as included.
20:         $Current\_Size \leftarrow$ current size of the calibration dataset.
21:         if $Current\_Size = (desired\_size \pm 2\%)$ then
22:             Break.
23:         else if $Current\_Size > (desired\_size + 2\%)$ then
24:             Mark the trajectory as excluded.
25:         end if
26:     end if
27:
28:     Find the next unexamined small trajectory.                   ▷ Upwards search
29:     if Found then
30:         Mark the trajectory as included.
31:         $Current\_Size \leftarrow$ current size of the calibration dataset.
32:         if $Current\_Size = (desired\_size \pm 2\%)$ then
33:             Break.
34:         else if $Current\_Size > (desired\_size + 2\%)$ then
35:             Mark the trajectory as excluded.
36:         end if
37:     end if
38: until (The calibration dataset is of the desired_size (±2%)) OR (There are no more unexamined trajectories)
39:
40: if (The calibration dataset is of the desired_size (±2%)) then
41:     $Error\_Code \leftarrow 0$
42: else
43:     $Error\_Code \leftarrow 1$
44: end if
45: return $Error\_Code$
```

mented using a record set of yearly flags that started active for all years, and gradually deactivated to reflect the temporal resolution of the remaining unexplored trajectories. Since this mechanism is part of secondary functions of the process, it was excluded from algorithm 3.1. Appendix C presents the detailed identification of the calibration dataset, along with the results of the temporal analysis of this dataset (in tabular form).

3.6 Data transformation.

As described in section 3.4, simple geographical intersections were used to match the main GPS data with the various digital maps described in section 3.2. In essence this is a transformation from the geographical dimension to a purely abstract classification. Apart from the various segmentations included in this section, no other transformations were applied to the original dataset.

3.6.1 Segmentations.

Segmentation of long trajectories originated from the hierarchical subdivision of complex problems [192], however, research standards require a consistent segmentation for all trajectories in the dataset.

According to the sampling theorem by Nyquist from 1928 [194], and proved to be theoretically and practically correct by Shannon in 1949 [223], the repetitive movement patterns of any species that can be detected by the acquisition of movement sampling data, in this case GPS position fixes, are limited to those patterns repeated at twice the time interval of the finest time scale instruments. In other words, if the best GPS collar used to monitor the movement of a grizzly bear was programmed to acquire a position every 20 minutes, the research will only be able to detect movement patterns repeated every 40 minutes.

Quality of results from studies that utilize geographically referenced data can be increased with consecutive analysis at different spatial scales, but it is better to limit the inference of global rules based on local-area results [213].

The following are the segmentations used in this work, and represent just a few of the possible segmentations for grizzly bear tracking datasets.

3.6.1.1 Annual and seasonal.

Multi-year trajectories were separated by year because of the large temporal discontinuity between the last point in one year and the first point in the subsequent year. This is supported by the annual biological period of grizzly bear hibernation (a.k.a denning) [40, 46, 58, 115, 183, 190].

3.6.1.2 Monthly.

Based on some of the behavioral studies in section 2.1 (e.g. [40, 76, 70, 130, 217]), and in an effort to reduce the processing time of all machine learning methods, the yearly trajectories were subdivided into monthly sections[*].

3.6.1.3 Weekly.

Given the dynamic nature of the straightforward trajectory similarity methods (section 2.4.1), a reduction on the individual trajectory segments was applied to reduce the overall execution time. With this finer scale approach it was also hoped that all machine learning methods in this work could identify a greater quantity of unique movement patterns in a reasonable amount of time.

3.6.1.4 Circadian and diel.

The proposed segmentation of this subsection responds to two main objectives, the simplification of geographic representation to speed up similarity calculations, and the conservation of topological relationships between trajectories and land cover areas.

[*]Julian calendar months, i.e. January, February, etc.

Including data acquired at different temporal scales into a single analytical tool requires careful consideration of the facts represented by each scale. The conclusions derived from such a tool will only be valid if the differences in the data are recognized and handled appropriately from the first analysis step.

Consider for a moment the matching of consecutive GPS positions acquired the same day of the year, and the categorical land cover map from section 3.4. Dividing the 24 hours of the day into regular intervals of the same size as their respective collar resolution will result in the conceptual temporal "coverage" shown in figure 3.8.a. In other words, this matching represents the type of habitat in which the bear is assumed to have stayed for the entire temporal period. Figure 3.8.b shows an alternative matching conceptualization with shorter time periods, of 20 minutes, based on the frequency distribution of the time intervals between consecutive trajectory positions, as shown in subsection 3.3.1.

Please note that the figure also shows periods of the day in which positions were not recorded. This is a natural result of the difficulty of acquiring GPS fixes for wildlife species and a limitation of the entire dataset.

Where more than one position fix was recorded in the same 20 minutes interval, the matching land cover classes determined the course of action. If all position fixes matched with the same land cover class, only the first position in the time period was considered for additional analysis (the rest were excluded from the final symbol concatenations described bellow). If the positions matched with different land cover classes, only the first position in the time period was considered for additional analysis for the following reasons.

The difference in the character of these decisions resides in the amount of information that the analysis might lose from each of them. In the first case, the land cover information is identical for each time period, then including multiple positions contributes no addition data for the analysis. The second case happens 2,237 times in the cleaned

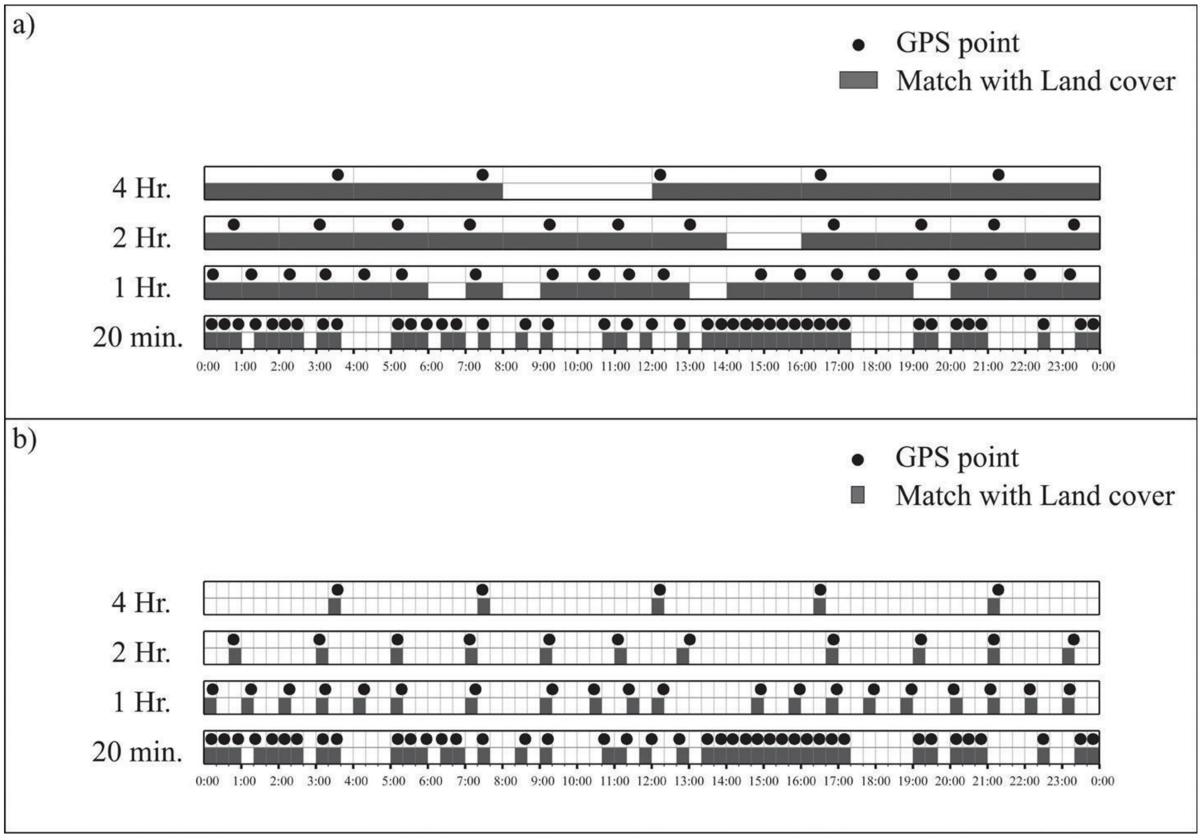

Figure 3.8: Circadian segmentation of multi-temporal resolution GPS data, matching with land cover map's classes a) at the original GPS resolution, b) at the resolution of the 20 minute interval data.

dataset for 9 different bears. The involved trajectories were analyzed to determine if the overall finer time period had to be changed, but their mean time interval was more than 23 minutes in each case. Comparing this inconsistencies with the total size of the cleaned dataset of 199,223 individual positions, the second case represents 1.12%, correspondonding to some of the special cases aluded to in section 3.3, and which were originally ignored in order to arrive at an initial general analysis tool.

Being strict in the use of the GPS data, each position fix occurs at a temporal instant. All other positions are unknown until the next GPS fix is acquired and can only be approximated*.

*Some "interpolation" methods to do this have already been mentioned in subsection 2.1.2

To allow the model to conduct a simultaneous multi-temporal analysis without having to divide every 24 hour period down to minute (second, millisecond, etc.) intervals, a reasonable compromise is to define "temporal coverage" as the period of the most detailed data in the dataset. In this case 20 minutes, as described in subsection 3.3.1 and illustrated in figure 3.8.b. This approach allows direct temporal comparison amongst all daily trajectories using only 72 pieces of information for each one.

Considering once more the alternative of dividing every 24 hours period into smaller temporal coverage periods, for example one minute each, every resulting daily trajectory would contain 1,440 pieces of information. Remembering that the straightforward similarity measurement methods perform in quadratic order (in the worst case), an increase from 72 to 1,440 pieces of information represents an increase in execution time from 5,148 comparisons to 2,073,600 comparisons per trajectory pair. Additionally, only 92 individual GPS positions in the entire cleaned dataset were acquired at ten minute intervals (or less), which would mean that for this version of the "completed" concatenation (as defined bellow), every segment would mostly contain "n/a" values. From the point of view of this work, the analysis of such data would take much longer than the proposed 20 minute temporal coverage alternative without adding any useful information either to the process or to the end results.

This matching and segmentation is the equivalent of assuming that the bear stays in the matched land cover terrain type for the entire 20 minutes period. This is an important limitation of this work and will be mentioned once again in the concluding chapter of this document.

As a direct consequence of this segmentation, the longest possible sequence consists of 72 consecutive GPS points, i.e. 3 fixes per hour, for 24 hours.

Analyzing the representation of the GPS points in terms of their matched abstract land cover classes, the daily sequences can be represented by concatenations of the abstract symbols in alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A\}$ (see table 3.2 for the class-land cover equivalences). When only land cover classes are involved, these concatenations can be seen as representations of bear transitions from one land cover to another (or the same class).

The same sequence can vary in its presentation depending on the three restrictive rules:

1. The consecutive repetition of the same symbol.
2. The exclusion of classes that represent absence of data (i.e. unknown values, $\sigma = \{0, 9, A\}$, "n/a", "cloud" and "shadow" respectively).
3. The level of attention given to the temporal dimension (i.e. the temporal "coverage" mentioned above).

From the possible combinations of these factors there are five representations of interest for the automatic analysis of abstract land cover transitions. With the correct restrictions, some of these concatenations can be defined as representing topological relationships of the individual trajectory with the different land cover areas. Other representations are possible, but this research judged the current ones as the most useful for automatic movement pattern detection.

I. Original. Ignore gaps in the temporal scale (allow rule 1, above).

II. Distinct. Original, but excluding consecutive symbol repetitions (restrict rule 1).

III. Artificial, repeated. Original minus "no data" symbols (rule 2).

IV. Artificial, distinct. Distinct minus "no data" symbols (II with rule 2).

V. Completed. Include all symbols. Add symbol "n/a" in every gap of the temporal scale (rule 3).

Table 3.3: Land cover class equivalents' concatenation, circadian segmentation.

Original	Distinct	Artificial, repeated	Artificial, distinct
535514115111111111115	535141515	535514115111111111115	535141515
4117711116111111115115151	417161515151	4117711116111111115115151	417161515151
1111111111111111111113	13	1111111111111111111113	13
0000000007003313113333	07031313	73313113333	731313
373373389331	373738931	37337338331	37373831
66A66666666666666A666AA66	6A6A6A6	6666666666666666666666	6

Table 3.4: Land cover class equivalents' concatenation, circadian segmentation (cont.).

Completed
500300500500100400100100500100100000100100100100100000100100100100500
400100100700700100100100100600100100100100100100500100100500100500100
100100100100100100100100100100100100000100100000000100100100100100300
000000000000000000000000700000000300300100300000000100100300300300000300
300000700000300000300000700000300000300000800000900000300000300000100000
600000600000A006006066006066006066006006006600A00000600606A0000A600600000

To clarify these combinations, tables 3.3 and 3.4 show examples of some of the concatenations present in the cleaned grizzly bear tracking dataset.

The distinct combinations (II and IV), will only concatenate different symbols next to each other, therefore, these are the closest representations to topological relationships.

3.7 Data mining.

In the context of this work, the term "machine learning" is better suited to describe the undertaken technical approaches, however, the term "data mining" is consistent with the KDD methodology proposed by Han and Kamber [128] (section 2.3), and kept for the rest of this document only where no ambiguity is possible.

3.7.1 Local alignment.

To help minimize the increase in execution time as the size of the analyzed datasets increase (i.e. scalability problems), as experienced with previous approaches (see appendix A), a hybrid local alignment method was considered for this analysis as presented in subsections 2.4.1.1 and 2.4.1.2. As compared to the original method developed for Biotechnology applications, the main departure of this work's method consists in the replacement of the original dynamic programming exploration of the set of candidate functional groups against the full molecule formulas (i.e. the matrix-based comparisons of every symbol in the functional groups against every symbol in the molecule sequences), by the exact matching of a set of relatively short subsequences. In other words, this hybrid approach was selected because, with its proposed structure it was reasonably expected to avoid most scalability and conceptual problems encountered with the methods implemented above.

The following subsection includes a brief description of the manual exploration process that allowed the conceptual combination of the theoretical details necessary to arrive at the combined analysis method presented in the rest of this document.

3.7.1.1 The initial conceptual diagram.

The process started with the idea of generating the production rules of the regular grammar that could describe the hierarchical tree defined by the single rule: "No consecutive repetitions of the same symbol". The first manual graphical assessment was conducted for a limited subtree with full failure links for all nodes, except the root and level-one nodes, which by definition do not have anywhere to fail to. The limitations of the tree consisted of a single starting symbol (0 in this case), three levels in total, and an alphabet of four symbols ($\Sigma = \{0, 1, 2, 3\}$).

Two tree expansions were conducted with full failure links for all nodes. Figure 3.9 shows a simplified diagram of the construction steps of these expansions. The first expansion introduced an additional starting symbol (1), and a level expansion to four

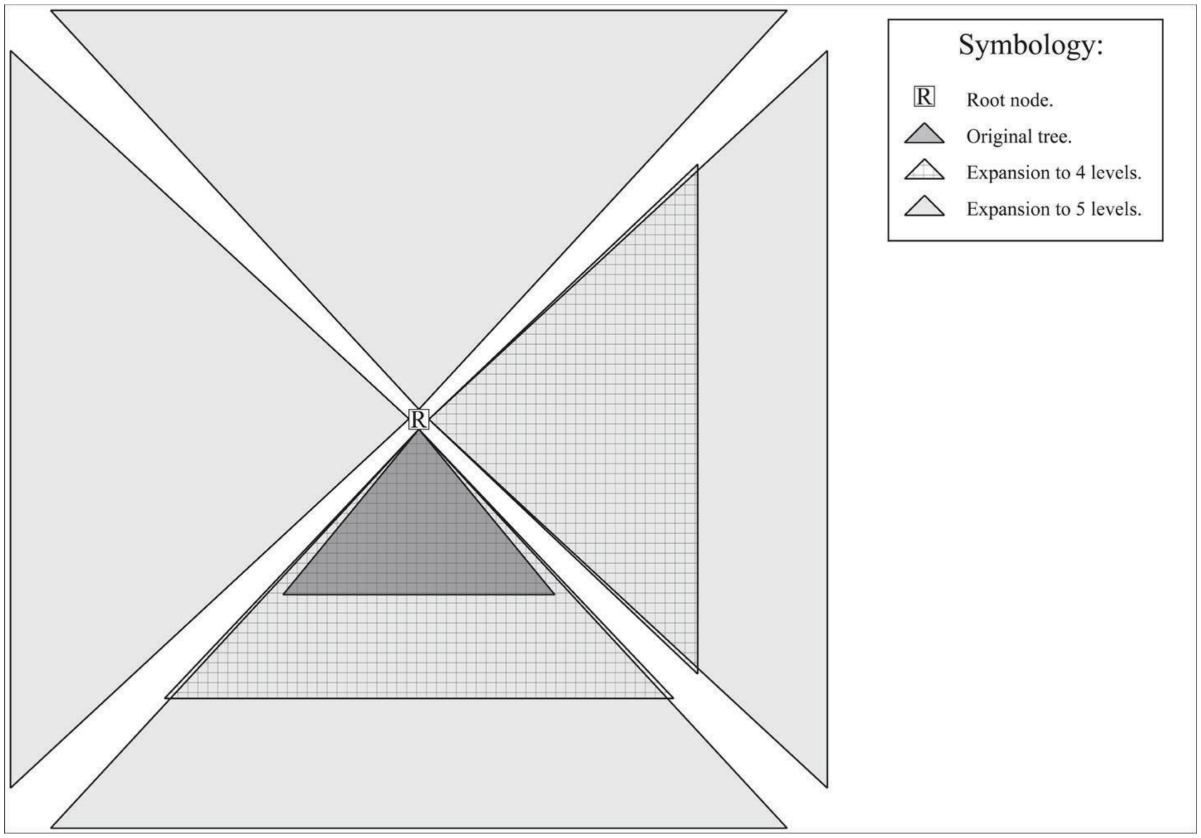

Figure 3.9: Construction of the initial conceptual diagram.

levels in both subtrees. Instead of drawing tree expansions in side-by-side subtrees as it is customary, the first expansion was constructed by drawing the corresponding subtree rotated 90^o counter-clockwise in relation to the original subtree. The second expansion introduced the remaining two starting symbols (2 and 3) and a level expansion of all subtrees to five levels in total. The expansion to five levels was constructed by drawing the necessary additional subtrees at consecutive 90^o counter-clockwise rotation angles from the second subtree introduced in the previous expansion.

The tree expansions can also be conceptualized as contiguously rotated triangular configurations as opposed to the customary parallel representation. Figure 3.10 shows this concept with complete analysis of its full failure links. In the judgment of the current project, this rotated contiguity shows the symmetrical behavior of the full failure links

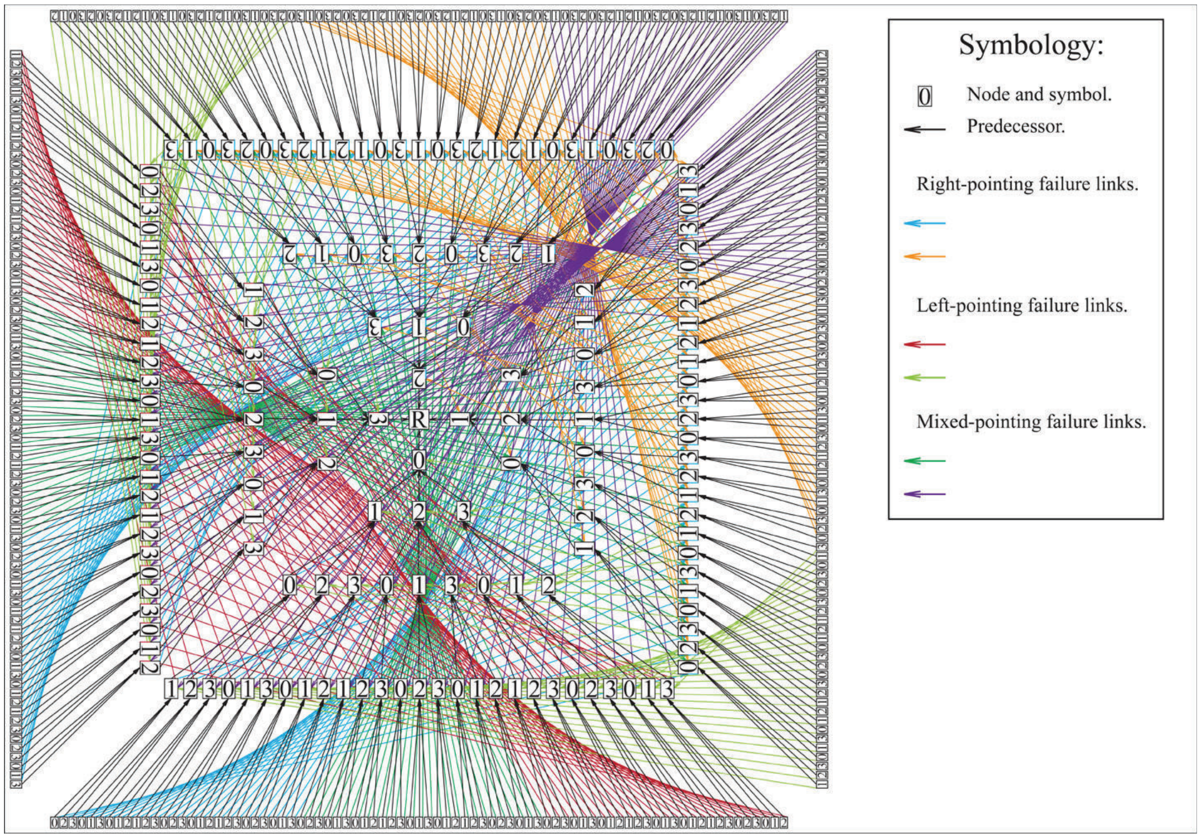
Symbology:
Node and symbol.
Predecessor.
Right-pointing failure links.
Left-pointing failure links.
Mixed-pointing failure links.

Figure 3.10: Initial conceptual diagram with full failure links, height = 5.

better than their corresponding side-by-side representation. Two other advantages of this representation include space economy, and direct visualization of tree rotations (useful for the conceptualization of tree matching processes).

Note that the symmetry axis for the full failure links in figure 3.10 (not shown in the figure), runs at an angle of 45^o from the upper right corner to the lower left corner of the square. The symmetry of the configuration is significant because, the equivalent data structure represented internally in any computer system could use these characteristics for detection of tree corruption and/or performance enhancements (e.g. partitions, supplementary indexes).

Table 3.5 shows the set of non-normalized production rules that describe the equiva-

Table 3.5: Production rules for the initial conceptual tree (I).

Production rules	Production rules (cont.)	Production rules (cont.)
$I_0 \rightarrow 0$	$I_{12} \rightarrow I_8 \mid I_9 \mid I_{10}$	$I_{24} \rightarrow I_0\ I_{23}$
$I_1 \rightarrow 1$	$I_{13} \rightarrow I_8 \mid I_9 \mid I_{11}$	$I_{25} \rightarrow I_1\ I_{22}$
$I_2 \rightarrow 2$	$I_{14} \rightarrow I_8 \mid I_{10} \mid I_{11}$	$I_{26} \rightarrow I_2\ I_{21}$
$I_3 \rightarrow 3$	$I_{15} \rightarrow I_9 \mid I_{10} \mid I_{11}$	$I_{27} \rightarrow I_3\ I_{20}$
$I_4 \rightarrow I_0 \mid I_1 \mid I_2$	$I_{16} \rightarrow I_0\ I_{15}$	$I_{28} \rightarrow I_{24} \mid I_{25} \mid I_{26}$
$I_5 \rightarrow I_0 \mid I_1 \mid I_3$	$I_{17} \rightarrow I_1\ I_{14}$	$I_{29} \rightarrow I_{24} \mid I_{25} \mid I_{27}$
$I_6 \rightarrow I_0 \mid I_2 \mid I_3$	$I_{18} \rightarrow I_2\ I_{13}$	$I_{30} \rightarrow I_{24} \mid I_{26} \mid I_{27}$
$I_7 \rightarrow I_1 \mid I_2 \mid I_3$	$I_{19} \rightarrow I_3\ I_{12}$	$I_{31} \rightarrow I_{25} \mid I_{26} \mid I_{27}$
$I_8 \rightarrow I_0\ I_7$	$I_{20} \rightarrow I_{16} \mid I_{17} \mid I_{18}$	$I_{32} \rightarrow I_0\ I_{31}$
$I_9 \rightarrow I_1\ I_6$	$I_{21} \rightarrow I_{16} \mid I_{17} \mid I_{19}$	$I_{33} \rightarrow I_1\ I_{30}$
$I_{10} \rightarrow I_2\ I_5$	$I_{22} \rightarrow I_{16} \mid I_{18} \mid I_{19}$	$I_{34} \rightarrow I_2\ I_{29}$
$I_{11} \rightarrow I_3\ I_4$	$I_{23} \rightarrow I_{17} \mid I_{18} \mid I_{19}$	$I_{35} \rightarrow I_3\ I_{28}$
		$I_{36} \rightarrow I_{32} \mid I_{33} \mid I_{34} \mid I_{35}$

lent regular grammar for this tree (I).

3.7.2 The proposed new algorithm.

Following extensive bibliographic research and from the experiences gained from the implementation of the LBS and simulated annealing methods, this work judged the conceptual combination of keyword tree structures with one of the fastest string matching algorithms [1, 23, 47, 48, 69, 123, 124, 137, 147, 148, 207, 231, 235], as a good alternative for automatic detection of movement patterns. Figure 3.11 shows the different parts of some previous methods that needed to be integrated for such an implementation.

From the Boyer-Moore [47, 48] method, the new algorithm adopted the concepts of examining the text from left to right and the pattern from right to left, with the purpose of maximizing the pattern slides. Combining this premise with the concept from the Aho-Corasick [1] method of simultaneous matching of a set of keywords, of any size, with the use of a hierarchical tree structure, the proposed algorithm implemented one of the fastest possible matching processes [1, 23, 47, 48, 69, 123, 124, 137, 147, 148, 207, 231, 235]. From the Apostolico-Giancarlo [23] method, the algorithm incorporated the suffix aware

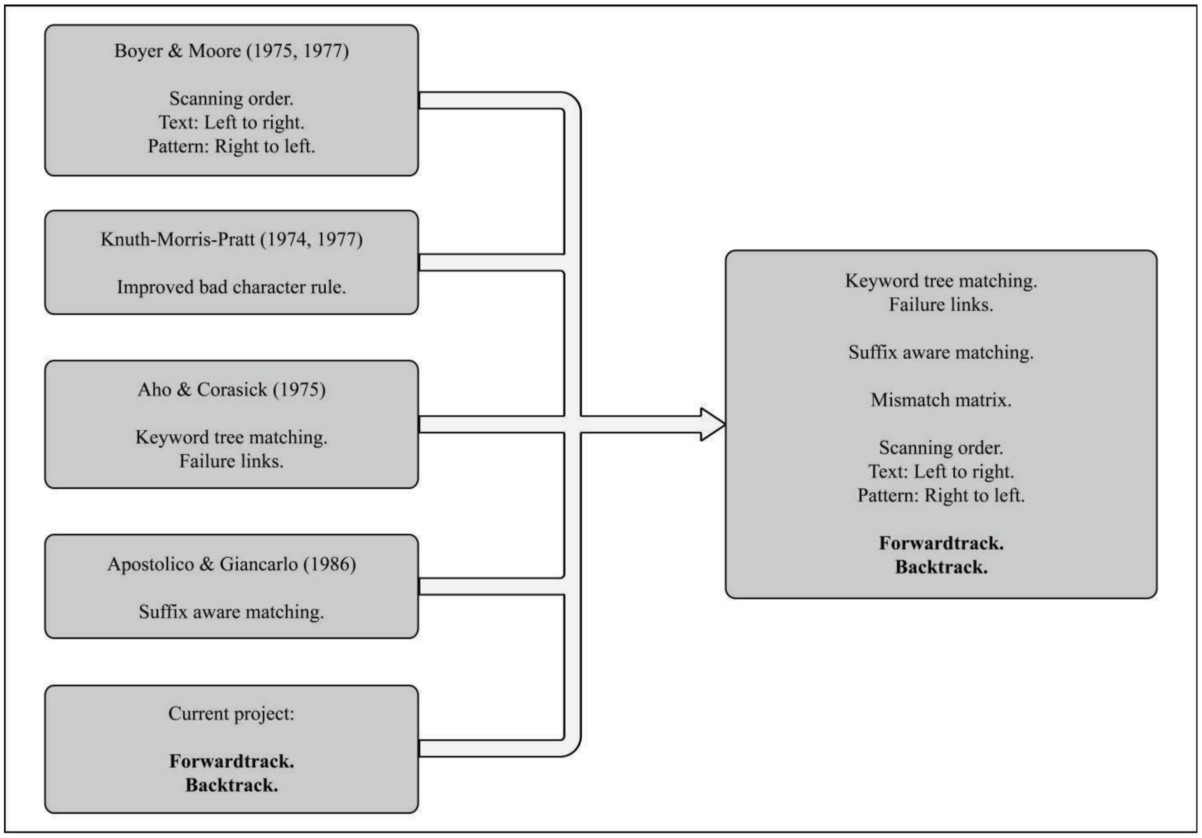

Figure 3.11: Key conceptual sources for the proposed new matching algorithm.

matching directly in its keyword tree structure, to reduce tree traversal as much as possible (i.e. the individual character examinations). To speed up tree traversal recovery and tree realignment after a character mismatch, the algorithm adapted the mismatch matrix structure and mechanism from the Knuth-Morris-Pratt [147, 148] method.

At the time of implementation, two other empirical areas of improvement were identified and incorporated, the tree's backtrack and forwardtrack, as described in subsection 3.7.2.1.

Summarizing, after an individual character mismatch, the main goal of this combination of methodological concepts is to preserve the maximum possible number of characters already matched. In other words, use the tree's own structure and its auxiliary structures, i.e. the list of full failure links and the corresponding mismatch matrix, to examine each text character as few times as possible.

3.7.2.1 Additional improvements.

Even after the combination of the theoretical studies mentioned in the previous subsection, the critical part of the algorithm, and the most time consuming, remains the realignment of the searched pattern with the text after a character mismatch. The current method adds the following empirical measures to improve the response time in such conditions:

1. **Forwardtrack.** [82, 146, 169, 220] Based on the knowledge of "the next character". For creation and exploration of failure links, and detection of "same character" subsequences.

2. **Backtrack.** [146, 169, 220, 229]Backwards tree traversal. To avoid recursive programming.

The concept of Forwardtrack will be further explained in subsection 3.7.2.5.

3.7.2.2 Design subsequences.

The first step in the implementation was to select a set of design subsequences that contained the largest possible repetition of suffixes without repeating the same full subsequence.

Table 3.6 shows the subsequences chosen for this work with two sets of Record Identifiers (Rids). The "Sorted Rid" corresponds to a regular alphabetical sorting, while "Reversed Rid" is assigned after an alphabetical sorting that analyzes the subsequence from right to left. This was done for clarity of the following analysis and to conform with the Boyer-Moore method (see figure 2.2).

Table 3.7 shows the same information, but with the subsequences written in reverse

Table 3.6: Sorted design subsequences.

Sorted Rid	Subsequence	Reversed Rid
1001	000000000	1
1002	000010000	4
1003	001002001	11
1004	001010200	8
1005	001032001	12
1006	013200120	10
1007	020010200	7
1008	032003020	9
1009	100000000	2
1010	100001000	6
1011	100010000	5
1012	100100000	3

Table 3.7: Reversed design subsequences (sorted).

Sorted Rid	Reversed subsequence	**Reversed Rid**
1001	000000000	1
1009	000000001	2
1012	000001001	3
1002	000010000	4
1011	000010001	5
1010	000100001	6
1007	002010020	7
1004	002010100	8
1008	020300230	9
1006	021002310	10
1003	100200100	11
1005	100230100	12

order. With the subsequences reversed, a regular alphabetical sort can be used to generate the order in this table.

The generation of the corresponding production rules for this set of subsequences is left to the reader [3, 4, 43, 51, 72, 79, 80, 103, 118, 134, 140, 141, 174, 175, 226, 230, 249].

3.7.2.3 Reversed keyword trees for the matching algorithm.

Using the subsequences of the previous two tables, two hierarchical search trees can

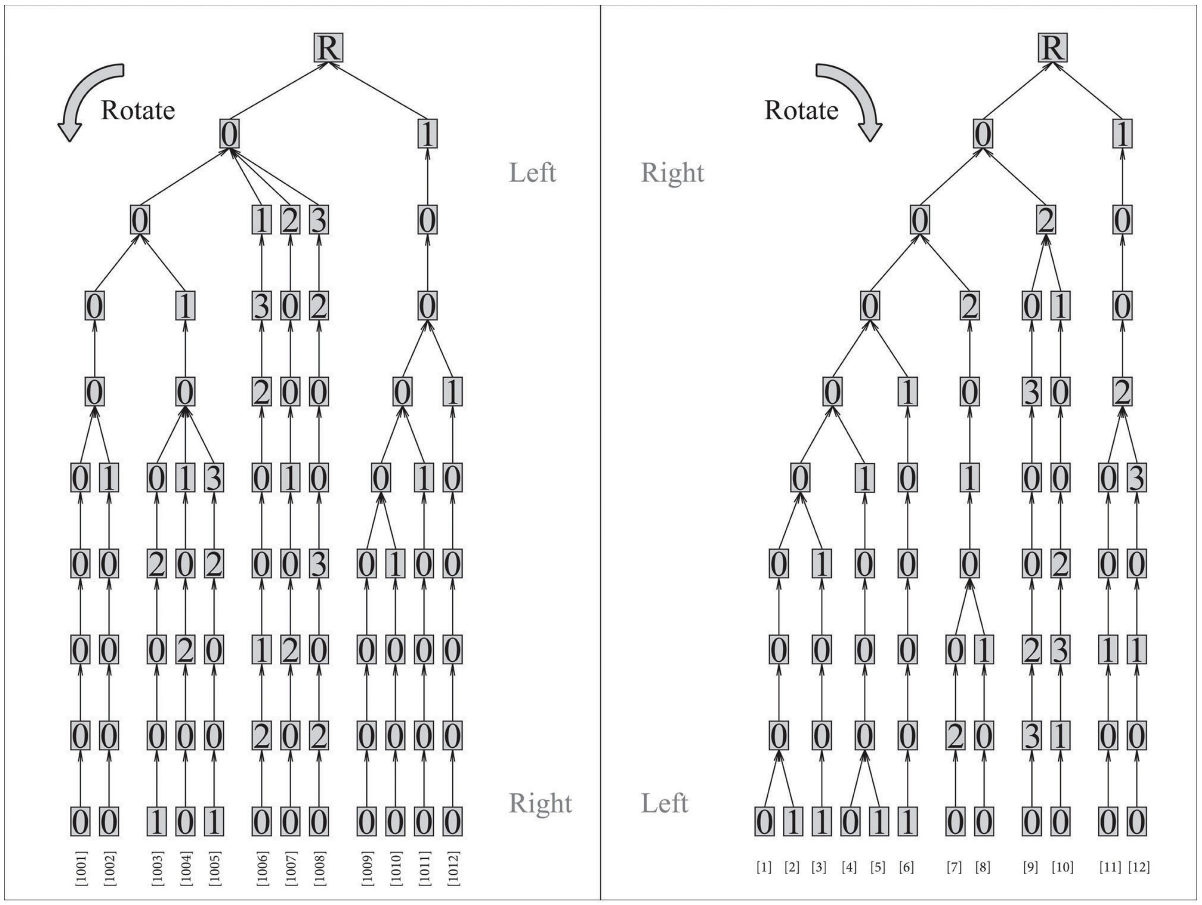

Figure 3.12: Keyword trees for the design subsequences, height = 9. a) Original keyword tree, and b) Reversed subsequence keyword tree.

be created, one character at a time, as shown in figure 3.12. Note the rotation arrows in both trees that will allow a better conceptualization of the matching of the tree with the text, and of the "sliding" of the tree (i.e. realignment). In these positions, the tree traversals of the rotated structures directly correspond to the individual character matching mechanisms of the search methods [1, 23, 47, 48, 147, 148]. Additionally, the tree sliding can be performed physically with the help of a printed copy of figure 3.12.

All remaining illustrations of keyword trees in this thesis adopt the rotated presentation.

Because the conceptual advantages of the tree rotation mentioned above, and the direction of the pattern examination of the Boyer-Moore method (i.e. from right to left),

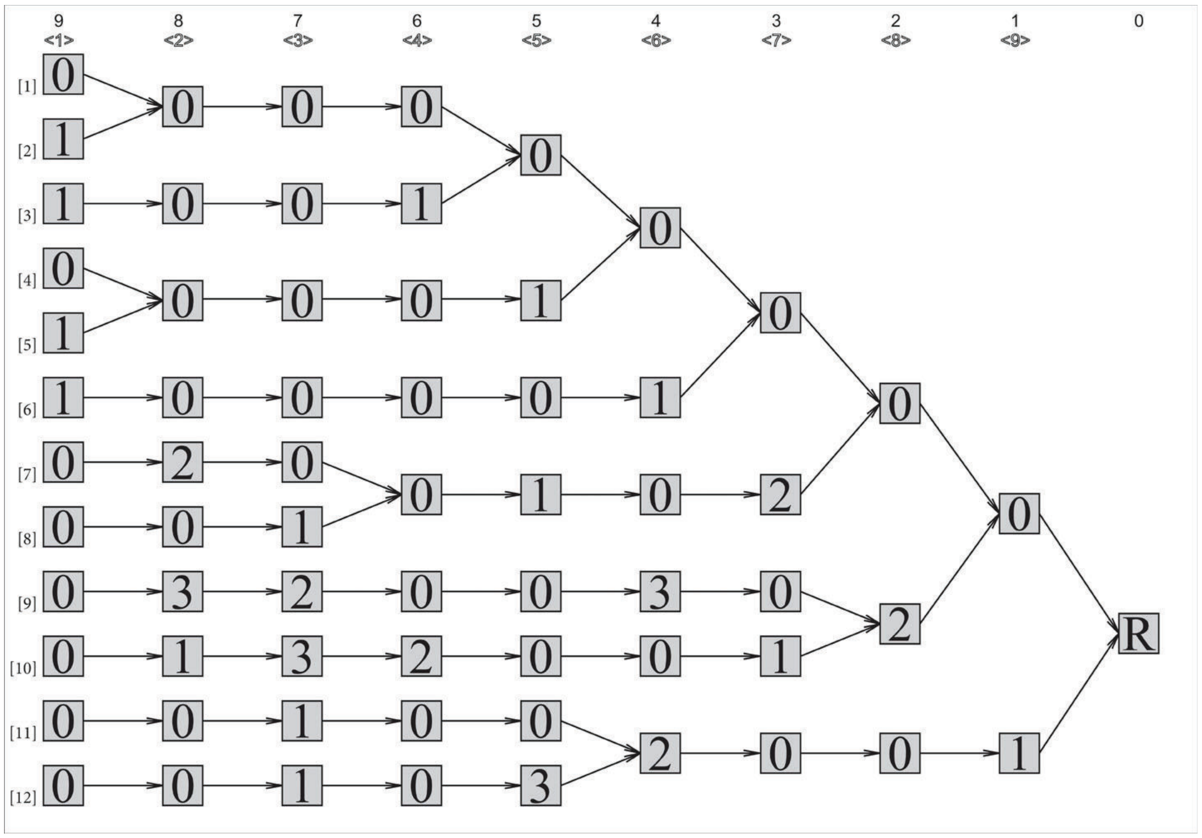

Figure 3.13: Reversed keyword tree for the design subsequences, height = 9.

the Reversed Keyword Tree (RKT) represents an adequate approach for this method. In other words, the traversal of the tree from right to left, as shown in figure 3.13, makes it conceptually (and geometrically) compatible with the Boyer-Moore method. Note the Arabic numerals between squared brackets placed at the far left of the figure. These numbers are the “Reversed Rid” values of tables 3.6 and 3.7, and will be present in all remaining illustrations of keyword trees in this document.

Figure 3.14 shows the same tree with additional structural details, whose utility will be explained momentarily. For now, let us just define the meaning of two of them (the rest are explained in the legend of the figure).

Attached to the upper-left corner of each node, a smaller rectangle shows the machine-assigned node identifier for the individual symbol. The rectangle attached to the bottom

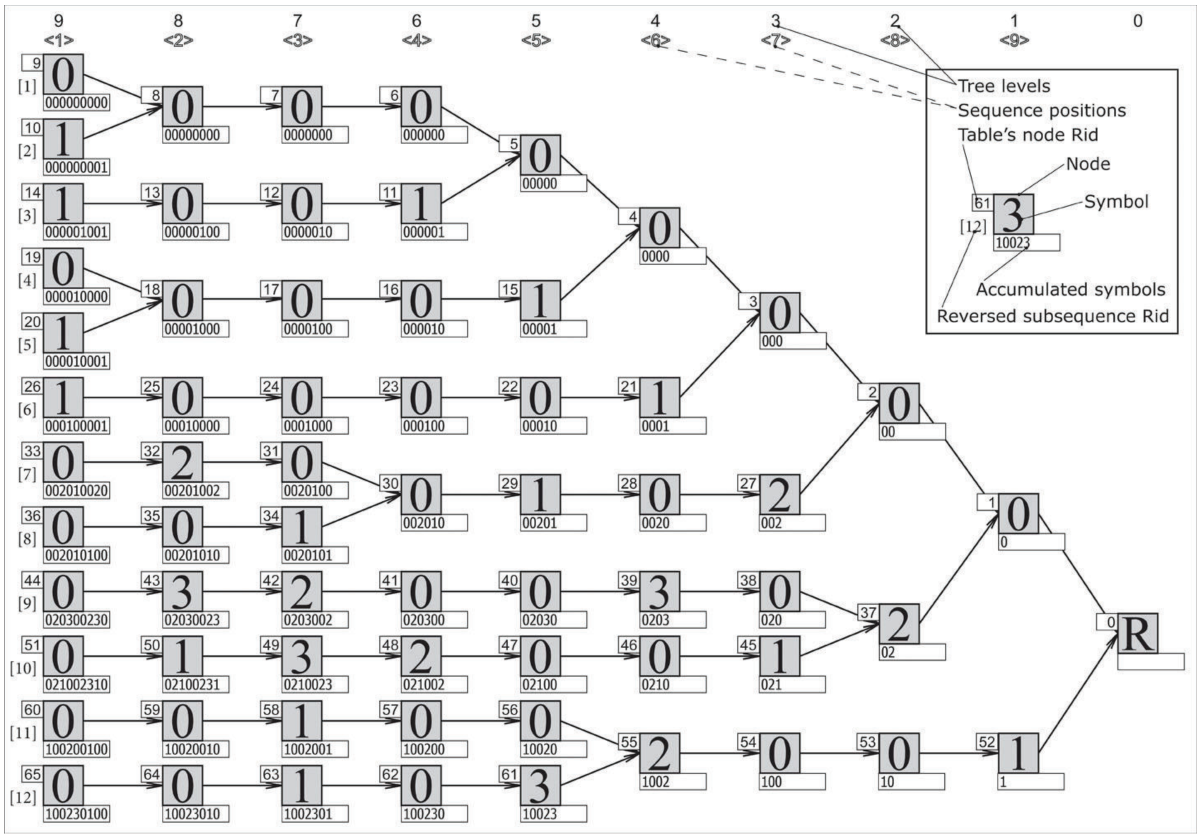

Figure 3.14: Reversed keyword tree for the design subsequences with table's Rids, and accumulated symbol concatenations.

of each node contains the accumulated concatenation of symbols resulting from traversing the tree from its root until reaching each node.

The failure links of all trees in subsection 2.4.1.3 slide the tree to the right of the text (imagine the trees in that subsection rotated as indicated in figure 3.12.a). For the reversed keyword tree, failure links calculated in the same manner would slide the tree to the left of the text, which is the opposite of what is needed for this method. Such links (right-pointing) can't be accepted because they imply one of two cases. If the matching has just started, such a link would slide the tree past the left end of the text, or, the tree has to slide to the left because the method missed a pattern in an earlier alignment.

Calculating the failure links for the reversed keyword tree in the opposite direction introduces the possibility of more than one valid link per node, and the need for additional storage structures for this list (potentially one list for each node in the tree).

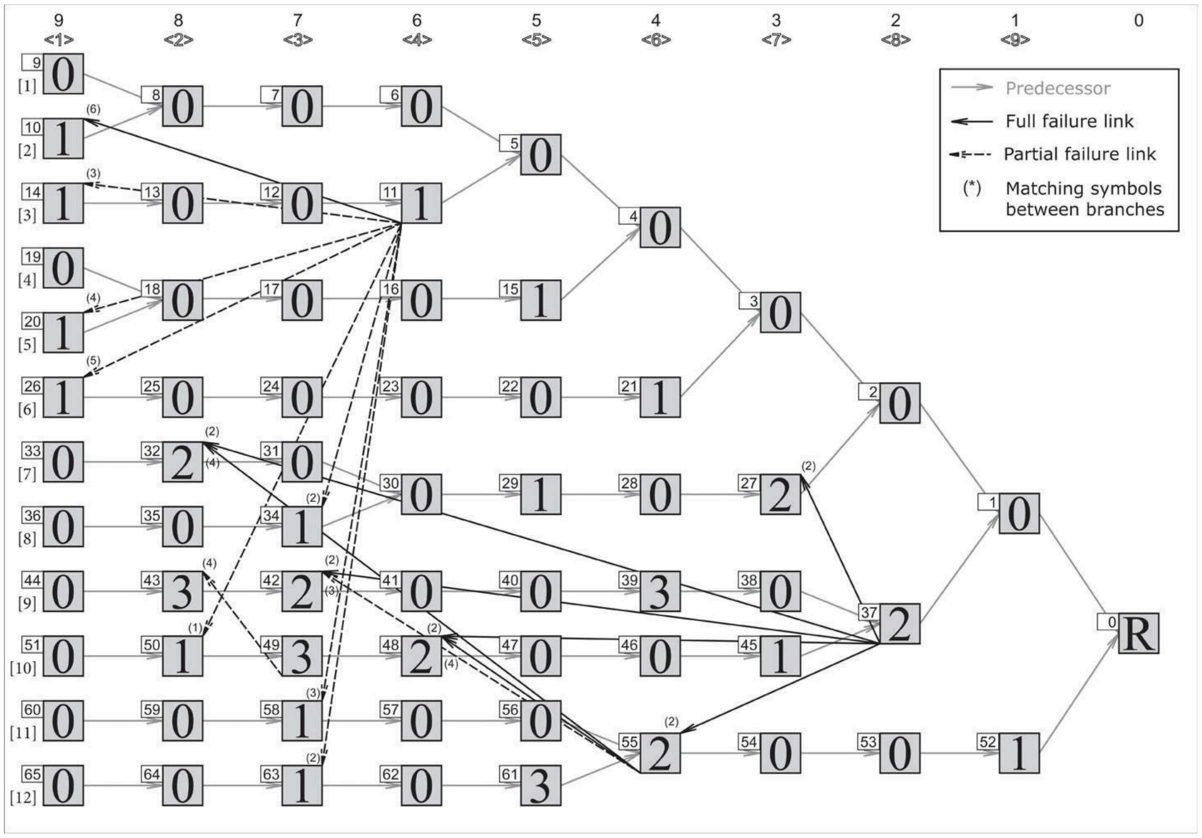

Figure 3.15: Left-pointing failure links for four example nodes (11, 37, 49, 55).

An example, with the assistance of figure 3.15, should clarify this concept. Let us imagine that the tree is aligned with the middle of the text* in such a way that the tree traversal has arrived at node 55 (reversed subsequence "1002"). If the next symbol in the text (to the left) is 1, then there will be a mismatch in this part of the tree (neither of the successor nodes of 55 contain a 1).

To preserve the maximum amount of work, the method has to look for alternative branches where the reversed subsequence "1002" is present, in this case, nodes 32 and 48. Both alternatives contain the entire reversed subsequence from node 55 in the exact same order, and can be considered valid "jump" points. Both of them are equally likely to

*I.e. far from any of its two ends.

allow the method to continue with a successful analysis, and from now on will be referred to as equivalent full failure links or equivalent failure links. However, in implementation terms, the node that is closer to the current node should be preferred. Distance in this type of tree is measured first by the difference of the tree levels between nodes, and then by the hierarchical position of the node (its absolute presentation order when the tree needs to be transformed into a two-dimensional graph). This definition of distance should be used to determine the absolute order in which exploration of the equivalent failure links will proceed.

Note that node 42 is a partial match to node 55 with the reversed subsequence “002”. Lets analyze what would happen if this partial failure link would be accepted in the list of valid “jump” points for node 55. First, lets assume that all other valid failure links for node 55 have already been explored and discarded, then node 42 is considered.

The jump from node 55 to node 42 has three symbols in common (“002”), therefore, the rest of the symbols to the right of node 40 in the branch containing node 42 have to be examined. This is the main use of the backtrack mechanism in this method. The symbol in node 39 is a 3, but this process already knew that the equivalent node in the original branch (node 52 in the branch that includes node 55) contains the symbol 1, which would produce an immediate mismatch. Since the process knew these facts from the moment the tree was constructed, it is useless to include partial links to the left of the current node in the list of valid failure links for any node. Since all right-pointing partial failure links had already being discarded, this statement discards partial failure links from this method altogether.

Apart from the sliding direction of the tree mentioned above, two more things have to be said about full failure links. First, no full failure links can occur at any tree level that is farther right than the current node. Since tree levels to the right of any given node contain less concatenated symbols, it is literally impossible to find a full failure link in any of those nodes. Finally, if a full failure link were to exist at the same tree level as the current node, because of the structure of the tree, it would have to be the current node itself. Since this process is interested in continuing the search after a character mismatch, "jumping in place" would only result in an infinite loop. In other words, no self joins should be allowed in terms of the list of equivalent full failure links.

This analysis implies that any other case of failure links for hierarchical keyword trees, full or partial, can be decomposed into a combination of the cases presented up to this point [3, 4, 51, 72, 79, 80, 103, 118, 134, 140, 141, 174, 175, 226].

Considering at the same time the concept of equivalent full failure links and node distance, as defined in the previous paragraphs, each node will have a best full failure link. This concept is illustrated in figure 3.16 for the example nodes, while figure 3.17 presents the same information for the entire tree.

In operational terms, this is almost irrelevant because once a mismatch has been detected, all alternative branches have to be considered until one is found to be adequate, or all alternatives have been exhausted. This means that a simple flag can take the place of the best equivalent full failure link in the data structure, but in terms of graphical presentation it is easier to understand a directed line between two nodes.

3.7.2.4 Mismatch matrix.

Adding the improvements from the Knuth-Morris-Pratt [147, 148] method to the current implementation took the form of a mismatch matrix, which is a completely independent structure from the keyword tree.

The role of the mismatch matrix is to determine the number of characters to slide the keyword tree, and the new tree cell, if it exists, at which the matching process can be restarted (discarding all previous matches, except the mismatch symbol). Once a

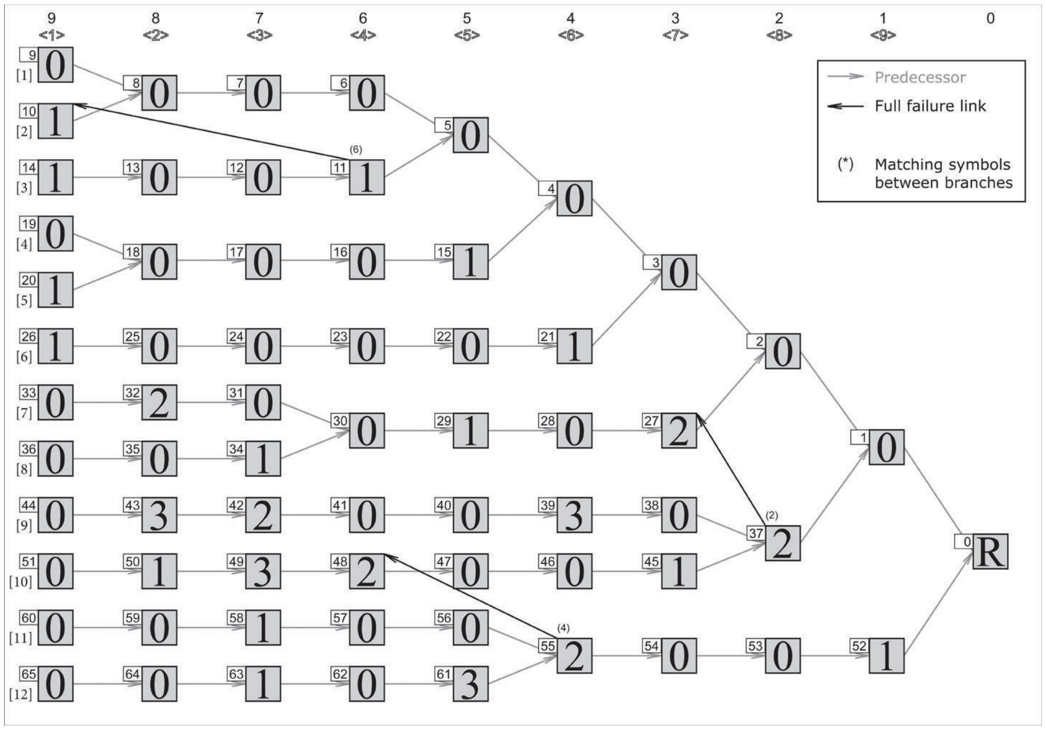

Figure 3.16: Best equivalent full failure links for four example nodes. No link of this type for node 49.

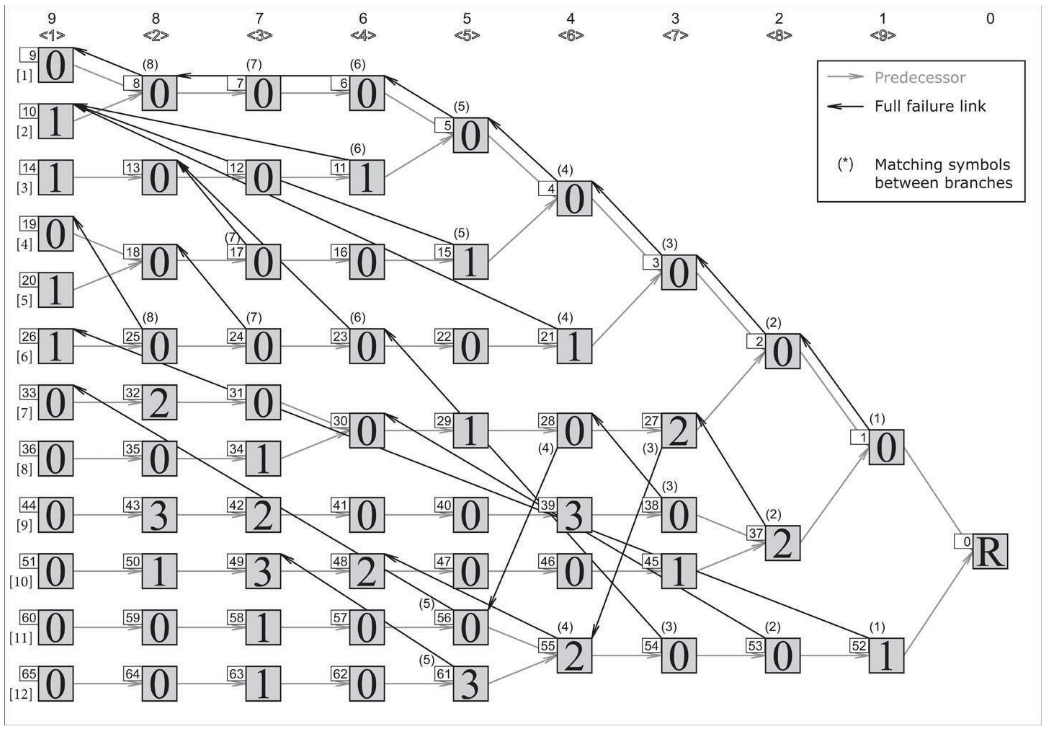

Figure 3.17: Reversed keyword tree for the design subsequences with best full failure links for all nodes.

Table 3.8: Bad character rule matrix, mismatch matrix, or slide matrix.

Tree level	**Symbol**										
(Pattern position)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**A**	**10**
1 (9)	0 /	0 /	1 37	3 39	9 /	9 /	9 /	9 /	9 /	9 /	9 /
2 (8)	0 /	1 45	0 /	2 39	8 /	8 /	8 /	8 /	8 /	8 /	8 /
3 (7)	0 /	0 /	0 /	1 39	7 /	7 /	7 /	7 /	7 /	7 /	7 /
4 (6)	0 /	0 /	0 /	0 /	6 /	6 /	6 /	6 /	6 /	6 /	6 /
5 (5)	0 /	0 /	1 48	0 /	5 /	5 /	5 /	5 /	5 /	5 /	5 /
6 (4)	0 /	0 /	0 /	1 49	4 /	4 /	4 /	4 /	4 /	4 /	4 /
7 (3)	0 /	0 /	0 /	0 /	3 /	3 /	3 /	3 /	3 /	3 /	3 /
8 (2)	0 /	0 /	0 /	0 /	2 /	2 /	2 /	2 /	2 /	2 /	2 /
9 (1)	0 /	0 /	1 /	1 /	1 /	1 /	1 /	1 /	1 /	1 /	1 /

character mismatch has been detected, and after verifying the incompatibility of all the equivalent full failure links of the corresponding tree node, the matching method finds the mismatch matrix cell corresponding to the combination of the tree level (conversely, the pattern position) and the mismatch symbol. Table 3.8 shows the mismatch matrix of the design RKT (figures 3.13 to 3.17) considering all the symbols of the original alphabet ($\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A\}$).

As an example, let us imagine that the traversal of the RKT in figure 3.17 has arrived to node 22. If the next symbol in the text were a “3” there would not be anywhere to jump in the tree to continue the matching process. In this case, the symbol “3” would have to be searched in the mismatch matrix (table 3.8), combined with the relative position in the text where it occurred (tree level 6 or pattern position 4). The corresponding cell in the matrix indicates that a slide of 1 position can be applied immediately to the entire process, and that the tree traversal can be restarted in backtrack mode from node 49.

For the symbols not included in the design subsequences (4, 5, 6, 7, 8, 9 and A), the corresponding tree slides are stored in the matrix cells and a NULL pointer (graphic symbol "/") denotes the restart of the tree traverse from the root.

This matrix is most effective for moderately dense to sparse keyword trees, combined with mid to large size alphabets [124, 147, 148].

3.7.2.5 Forwardtrack.

This work defines *Forwardtrack* as an empirical optimization principle based on the concept of "the next symbol" [82, 146, 169, 220].

To reduce the size of the result set of any combinational query, it employs the next symbol (from the current node) and the equal or "not equal" relationship. It was used in the following parts of the matching algorithm (illustrated in T-SQL code):

- Creation of the full failure links for each node (algorithm 3.2).
- Loading of the equivalent full failure links that have the mismatch symbol as their "next symbol" (algorithm 3.3).
- Detection of consecutive overlapping subsequences composed entirely of repetitions of the same symbol (e.g. "555555555", algorithm 3.4).

3.7.2.6 Matching algorithm.

To meet the objective of a minimalistic approach, and considering once more that standard ANSI / ISO / IEC SQL-2010 does not provide integrated support for complex or user-defined data types [13, 14, 15, 16, 17, 18, 19, 20, 21, 22], the subsequence matching

Algorithm 3.2 Creation of equivalent full failure links with Forwardtrack, SQL

```
SELECT Tgt.Rid, Tgt.Tree_Level
FROM FROM dbo.Reverse_KeyTree_Orig_Asy As Tgt
INNER JOIN dbo.Reverse_KeyTree_Orig_Asy As Orig_Suc          ▷ Successors of the current node.
ON Orig_Suc.Parent_Rid = @Current_Node_Rid
LEFT OUTER JOIN dbo.Reverse_KeyTree_Orig_Asy As Tgt_Suc           ▷ Targets' successors.
ON Tgt.Rid = Tgt_Suc.Parent_Rid
WHERE Tgt.Tree_Level = @Exploring_Tree_Level
AND Tgt.Rid <> @Current_Node_Rid                                   ▷ No self-joins.
AND Substring(Tgt.Sub_Sequence, (Tgt.Tree_Level−@Current_Node_Tree_Level+1),
                    @Current_Node_Tree_Level) = @Current_Node_SubSequence
AND isNull(Tgt_Suc.Symbol,−1) <> Orig_Suc.Symbol                   ▷ No equal successors.
GROUP BY Tgt.Rid, Tgt.Tree_Level
ORDER BY Tgt.Rid;
```

Algorithm 3.3 Loading of equivalent full failure links with Forwardtrack, SQL

```
SELECT Eq.Target_Rid, Eq.Matching_Symbols,
          (Eq.Target_Tree_Level − Eq.Source_Tree_Level) As Tree_Level_Diff
FROM dbo.Equivalent_Links_Orig_Asy As Eq
LEFT OUTER JOIN dbo.Reverse_KeyTree_Orig_Asy As Tgt_Suc
ON Eq.Target_Rid = Tgt_Suc.Parent_Rid                              ▷ Targets' successors.
WHERE Eq.Source_Rid = @Current_Node_Rid
AND isNull(Tgt_Suc.Symbol, @Mismatch_Text_Symbol) = @Mismatch_Text_Symbol
ORDER BY 2 DESC, 3, 1;
```

Algorithm 3.4 Same character detection. Forwardtrack, SQL

```
SET @Current_Text_Position = @Current_Text_Position + 1;

SELECT @Same_Symbol_Flag =(CASE isNull(Rid, −1) WHEN -1 THEN 0 ELSE 1 END)
FROM dbo.Reverse_KeyTree_Orig_Asy
WHERE Rid = @Current_Node_Rid
AND Sub_Sequence = Replicate(@Text_Symbol, @Keyword_Max_Len);

SET @Full_Seq_Symbol = @Text_Symbol;
WHILE ((@Same_Symbol_Flag = 1)AND(@Current_Text_Position <= @Text_Len)) BEGIN
    SET @Current_Symbol = Substring(@Text_Sequence, @Current_Text_Position, 1);

    IF (@Current_Symbol = @Full_Seq_Symbol) BEGIN
        INSERT INTO dbo.Matches_Orig_Asy(Tree_Rid, Text_Rid, Text_End_Position)
            VALUES(@Current_Node_Rid, @Text_Rid, @Current_Text_Position);

        SET @Current_Text_Position = @Current_Text_Position + 1;
    END ELSE BEGIN
        SET @Same_Symbol_Flag = 0;
    END;
END;
```

Algorithm 3.5 *Examine_FLinks*

```
Input: $EquivalentFLinks$, $NumIgnoreSymbols$
Output: $FailureLinkFlag$
 1: $FailureLinkFlag \leftarrow 0$
 2: while ((FailureLinkFlag = 0) AND (NOT End $EquivalentFLinks$)) do
 3:     $\boxed{\text{Traverse backward } ReversedKeywordTree(NumIgnoreSymbols)}$    ▷ Ignore these.
 4:     while ((FailureLinkFlag = 0) AND (NOT at Root node)) do
 5:         $\boxed{SymbolMatch \leftarrow \text{Traverse backward } ReversedKeywordTree}$    ▷ Backtrack.
 6:         if ((SymbolMatch = 1) AND (At Root node)) then
 7:             $FailureLinkFlag \leftarrow 1$
 8:         end if
 9:     end while
10: end while
```

Algorithm 3.6 *Examine_Earlier_FLinks*

```
Input: $EquivalentFLinks$, $NumIgnoreSymbols$
Output: $FailureLinkFlag$, $MismatchShiftDistance$
 1: $FailureLinkFlag \leftarrow 0$
 2: while ((FailureLinkFlag = 0) AND (NOT at Root node)) do
 3:     $MismatchShiftDistance \leftarrow MismatchShiftDistance + 1$
 4:     $\boxed{\text{Traverse backward } ReversedKeywordTree}$    ▷ Backtrack.
 5:     $\boxed{\text{Load } EquivalentFLinks}$    ▷ From new current node.
 6:     $FailureLinkFlag \leftarrow$ Examine_FLinks($EquivalentFLinks$)
 7: end while
```

method had to be implemented as an iterative program without complex RAM data structures.

Recursion was also consciously avoided because of all the unknown factors in its use inside a relational database system. As a consequence, the final program is long and relatively complicated, with several nested loops for different tasks and a minimum of sub procedure calls (algorithms 3.5, 3.6 and 3.7).

3.7.2.7 Computational complexity.

There are three major components that determine the space requirements for this method, the Reversed Keyword Tree (RKT), the lists of equivalent failure links, and the

Algorithm 3.7 *Dictionary_Exact_Matches*

```
Input: $Text$, $ReversedKeywordTree$, $MismatchMatrix$, $EquivalentFLinks$
Output: $KeywordMatches$
 1: repeat
 2:     $\boxed{CurrentTextSection \leftarrow \text{Load next } Text \text{ section}}$
 3:     repeat
 4:         $MismatchFlag \leftarrow 0$
 5:         $MismatchShiftDistance \leftarrow 1$
 6:         $CurrentKeywordPosition \leftarrow MaxKeywordLength$
 7:         while ((No Keyword match detected) AND (NOT End $CurrentTextSection$)) do
 8:             $\boxed{MismatchFlag \leftarrow \text{Traverse forward } ReversedKeywordTree}$
 9:             if (MismatchFlag = 0) then                                  ▷ Symbol match.
10:                 $NumIgnoreSymbols \leftarrow NumIgnoreSymbols + 1$
11:                 $CurrentKeywordPosition \leftarrow CurrentKeywordPosition - 1$
12:             else                                                        ▷ Symbol mismatch.
13:                 $\boxed{MismatchShiftDistance \leftarrow \text{Retrieve } MismatchMatrix \text{ value}}$
14:                 if (MismatchShiftDistance = 0) then
15:                     $FailureLinkFlag \leftarrow 0$
16:                     if (Valid link at current node) then                ▷ Forwardtrack.
17:                         $\boxed{\text{Load } EquivalentFLinks \text{ with Forwardtrack}}$
18:                         $FailureLinkFlag \leftarrow$ Examine_FLinks($EquivalentFLinks$)
19:                         if (FailureLinkFlag = 0) then                   ▷ Backtrack.
20:                             $FailureLinkFlag \leftarrow$ Examine_Earlier_FLinks($EquivalentFLinks$)
21:                             if (FailureLinkFlag = 0) then               ▷ All links explored.
22:                                 $MismatchShiftDistance \leftarrow MaxKeywordLength$
23:                             end if
24:                         end if
25:                     else                                                ▷ Backtrack.
26:                         $FailureLinkFlag \leftarrow$ Examine_Earlier_FLinks($EquivalentFLinks$)
27:                         if (FailureLinkFlag = 0) then                   ▷ All links explored.
28:                             $MismatchShiftDistance \leftarrow MaxKeywordLength$
29:                         end if
30:                     end if
31:
32:                     if (FailureLinkFlag = 1) then       ▷ Successful jump to different tree branch.
33:                         $CurrentTextPosition \leftarrow CurrentTextPosition + MismatchShiftDistance$
34:                         $\boxed{\text{Adjust all other counters and pointers.}}$
35:                     end if
36:                 end if
37:             end if
38:         end while
39:
40:         if (Keyword match detected) then
41:             $\boxed{\text{Register match in } KeywordMatches}$
42:             $CurrentTextPosition \leftarrow CurrentTextPosition + 1$
43:             if (Matched keyword is all the same symbol) then                ▷ Forwardtrack
44:                 $SeqSymbol \leftarrow$ Current symbol of matched keyword
45:                 while ((Same symbol at $CurrentTextPosition$) AND (NOT End $CurrentTextSection$)) do
46:                     $\boxed{\text{Register match in } KeywordMatches}$
47:                     $CurrentTextPosition \leftarrow CurrentTextPosition + 1$
48:                 end while
49:             end if
50:         else
51:             $CurrentTextPosition \leftarrow CurrentTextPosition + MismatchShiftDistance$
52:         end if
53:     until (All $CurrentTextSection$ has been examined)
54: until (All $Text$ has been examined)
```

mismatch matrix.

The space required to store the RKT is given by the size of the alphabet ($\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A\}$; $\mid \Sigma \mid \; = \; l$), the number of symbols in the longest subsequence (n), and the number of subsequences to search for (β), and has an upper bound of $O(l^n)$.

The amount of space required to store all the lists of full failure links depends on the number of symbols to look for (from one level to another), the tree density (i.e. keyword density), and the length of the longest subsequence (n). For one symbol at tree level one, it has an upper bound of $O(l^n)$. For two symbols at level two, it has an upper bound of $O((n-2)l^{n-2})$, etc[*]. In practice however, it would be very rare to encounter a tree so dense that the upper bound would be reached.

The mismatch matrix (subsection 3.7.2.4), has a fixed space requirement of $O(n * l)$.

3.8 Pattern evaluation.

The most frequent subsequences at the end of the matching process, presented in subsection 3.7.2.6, were clustered using two different algorithms.

The first approach followed the improved method in [221], first defined by Altschul et al. in [8]. The fundamental concept of the molecule formula's (i.e. text segments) clustering, groups all those molecules that contain several non-overlapping subsequences from the searched set. Because of their original dynamic programming implementation, the concept is referred to as the detection of molecules with multiple non-overlapping subsequence hits contained in the same matrix diagonal. The equivalent in the case of this work is simply the trajectory segments that contain multiple non-overlapping subsequences.

In essence, this first clustering method highlights the trajectory segments that contain

[*]The rest of this analysis is of combinatorial nature [25, 117], and falls outside the scope of this document.

similar structural subsequences. However, this fact might appear clearer when the results of the clustering are presented in two dimensional graphical form (i.e. when identified in the geographic maps of the corresponding trajectories). This approach was applied across all the analyzed subsequence sizes, returning similar results for all sensitivity levels with one special provision, the relevance of the different trajectory segments was relatively maintained as the subsequence size increased. For example, a trajectory segment of 70 symbols, contained four subsequences of 15 symbols each, while the same trajectory segment contained only two subsequences of 30 symbols each. Further details of the result of this clustering technique will be discussed in subsection 4.2.6.

For the second clustering approach, this work used a slightly modified implementation of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method, as described in the text by Han and Kamber ([128], pg. 418). The current work substituted the original concept of the "point in space" for the corresponding centroid of the two dimensional Minimum Binding Rectangle (MBR) of each of the matched subsequences (subsection 3.7.2.6). Otherwise, the process was allowed to cluster the subsequences as originally designed, varying only the ε and $MinPts$ parameters.

As with the first clustering approach described above, the results from this clustering method were compared across sensitivity levels to determine their relevance. Some subsequences were identified as similar in terms of their geographic location, and some of their geographic uses of space, but as it is expected of a geometrically-aware method, its results might be better explained when presented in map form (as will be discussed in subsection 4.2.6).

3.9 Knowledge presentation.

After the matching process, the most frequent subsequences of all sizes were plotted on top of the original land cover raster. This approach proved to be adequate for further visual exploration, as well as an alternative to reflect the geographic nature of the original dataset and its supplementary data sources. Furthermore, the results of the clustering methods described in the previous section might be better analyzed in a two dimensional representation, as contrasted with a pure numerical evaluation (e.g. mean distance between cluster centroids, mean distance between individual GPS positions and the edge of their corresponding vegetation patch).

The different nature of the interpretations of the purely tabular results (i.e. numeric), compared with their geographic equivalents, seem to support the value and relevance of the decision to present the results of the current work in bidimensional representation. In practical terms however, this representation is limited by the geographic extension of the study area, and might only be effective as a set of computer files fit for manipulation and further exploration by other researchers, with tools such as ESRI ArcCatalog 9.3. Alternatively, these results could be presented with the use of interactive multimedia representations.

3.10 Chapter summary.

This chapter presented the detailed application of the process known as KDD, slightly modified from the methodology proposed by Han and Kamber [128] (subsection 2.3), applied to a grizzly bear tracking dataset.

The chapter started with a description of the overall application approach for the suitable method in this work (section 3.1).

Section 3.2 described the study area, as well as the main dataset and the supplementary data sources used in this analysis, including a brief description of the dominant vegetation of the natural regions included in the study area.

To present consistent data to the machine learning methods, cleaning tasks were undertaken and described in sections 3.3, 3.4 and 3.5. Subsection 3.3.1 analyzed the distribution of the temporal intervals between consecutive pairs of trajectory positions of the cleaned dataset, to define its maximum conceptual temporal resolution (20 minutes).

The following section (3.6) centered on some appropriate segmentations for grizzly bear movement that allowed an overall reduction of complexity for the analysis tasks. In this case, a reduction on the size of the movement sequences presented to the mining methods ensuring at the same time that topological relationships and statistically significant details were not lost because of this reduction.

Ignoring the concept of time context, section A.1.1 presented the details of an LBS method intended to automatically detect trajectory segments containing multiple similar subsequences.

The data mining methods in this work, employed two solutions, a simulated annealing algorithm (section A.2), and a local alignment approach (section 3.7.1). One of the aspects of the local alignment method motivated an extensive step-by-step geometric analysis of the use of failure links for keyword trees employed in the exact matching searches included in this work (3.7.2.3). The section paid particular attention to the modification of previous string matching and amino acid sequence searching methods that were combined to generate the current hybrid program (3.7.2), including its main algorithmic expression (3.7.2.6).

The chapter closed with a brief description of the pattern evaluation and knowledge presentation decisions adopted for analysis of results, sections 3.8 and 3.9 respectively.

In order to fulfill its research objectives (section 1.3), the current chapter included detailed analysis of theoretical and empirical aspects of potential analysis algorithms for the automatic detection of frequent patterns from movement tracking datasets (appendix A, and section 3.7). The included details explored the technical features and flexibility of the candidate methods, their level of compliance to the ANSI / ISO / IEC SQL specifications [13, 14, 15, 16, 17, 18, 19, 20, 21, 22] standards, and the detailed graphical representation of the local alignment approach. Even when the methods included in appendix A where ultimately determined unsuitable for the effective analysis of the grizzly bear dataset, their successful fully integrated implementation advanced the notions of robustness and maturity of contemporary RDBMS platforms.

Additionally, other general aspects of the research were presented to identify necessary preparatory or subsequent tasks for the analysis, such as data cleaning, data integration, data segmentation, pattern evaluation and knowledge presentation, as well as a general description of where the machine learning methods fit in terms of the overall analysis process.

Chapter 4

Results and analysis

Although, in their present formulation, the LBS and the simulated annealing approaches are considered unsuitable for the effective analysis of grizzly bear tracking data, their fully integrated implementation serves to support RDBMSs as robust and mature, therefore, viable solution platforms. The implementation details of both methods are included in this document as peripherally relevant material in appendix A.

At this point in the project, the only implemented method that is considered suitable for the effective analysis of grizzly bear tracking data is the local alignment approach. The rest of this chapter presents and discusses averaged results of ten executions of the local alignment method, rounded to the nearest integer.

4.1 Benchmark platform.

All experiments in this work were conducted using a desktop computer equipped with an Intel Core 2 Quad processor at 3.00 GHz, with 8 GB of RAM, running Microsoft Windows XP Professional (2002) with service pack 3, and Microsoft SQL Server 2005 Express edition.

4.2 Local alignment.

Initial implementation of this method was conducted for multiple keywords of nine symbols in length (subsequences, $n = 9$) using the overall process detailed in section 3.1.

Once the unit testing of the implementation was finished, five daily segmentation alternatives (subsection 3.6.1.4) were combined with two versions of the log-likelihood ranking matrix (subsection 2.4.1.2) to generate ten different candidate subsequence sets.

- Original segmentation, asymmetric log-likelihood ranking. (Orig_Asy)
- Original segmentation, symmetric log-likelihood ranking. (Orig_Sym)
- Distinct segmentation, asymmetric log-likelihood ranking. (Dist_Asy)
- Distinct segmentation, symmetric log-likelihood ranking. (Dist_Sym)
- Artificial, repeated segmentation, asymmetric log-likelihood ranking. (ArtR_Asy)
- Artificial, repeated segmentation, symmetric log-likelihood ranking. (ArtR_Sym)
- Artificial, distinct segmentation, asymmetric log-likelihood ranking. (ArtD_Asy)
- Artificial, distinct segmentation, symmetric log-likelihood ranking. (ArtD_Sym)
- Completed segmentation, asymmetric log-likelihood ranking. (Comp_Asy)
- Completed segmentation, symmetric log-likelihood ranking. (Comp_Sym)

In turn, these candidate sets were used to construct the corresponding ten versions of the RKTs with all other auxiliary structures (henceforward referred to as tree types).

For each segmentation and log-likelihood ranking combination, i.e. for each tree type, subsequences of 3, 6, 9, 12, 15, 20, 25, 30 and 40 symbols in length were processed to assess the level of multi scale sensitivity of the method. Because of the characteristics of the diel segmentations of the data (subsection 3.6.1.4), except for the case of the "completed" sub variations, these subsequence lengths do not reflect an actual time interval. However, these were deemed adequate for this analysis because they cover, from a small section, to more than half of the maximum 72 symbols per daily segment. Making them uniformly applicable for all segmentations.

4.2.1 Memory and secondary storage space requirements.

For this subsection, any use of the term “space” without any other qualifier refers to computer RAM and secondary storage space, not to the concept of geographic or three dimensional space.

Disregarding the pagination needs of the RDBMS, there are three major components that determine the space requirements for this method; the Reversed Keyword Tree (RKT), the lists of equivalent failure links, and the mismatch matrix.

The space required to store the RKT is given by the size of the alphabet ($\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A\}$; $\mid \Sigma \mid \; = l$), the number of symbols in the longest subsequence (n), and the number of subsequences to search for (β), and has an upper bound of $O(l^n)$.

The amount of space required to store all the lists of full failure links depends on the number of symbols to look for (from one level to another), the tree density[*], and the total number of levels in the tree (n, the length of the longest subsequence). For one symbol at tree level one, it has an upper bound of $O(l^n)$. For two symbols at level two, it has an upper bound of $O((n-2)l^{n-2})$, etc[†]. In practice however, it would be very rare to encounter a tree so dense that the upper bound would be reached.

As an example of such empirical conditions, table 4.1 shows the number of nodes per level in the reversed keyword tree that have at least one valid full failure link at a higher tree level, and the total number of equivalent full failure links per tree level (stored in a different database table).

Figure 4.1 shows the same information in graphical form for the nodes in the tree with at least one valid full failure link, while figure 4.2 shows the total number of equivalent full failure links in the associated table.

The second figure, 4.2, shows a decreasing number of full failure links as the number of

[*]Which is the same as the keyword density.
[†]The rest of this analysis is of combinatorial nature, and falls outside the scope of this document.

Table 4.1: Full failure links by number of matching symbols, original asymmetric tree, n = 9.

	Tree		**Equivalent**
Symbols	All records	Full failure links	Full failure links
1	11	10	5,138
2	53	50	4,954
3	158	135	4,469
4	323	244	3,624
5	540	344	2,557
6	769	360	1,520
7	954	329	830
8	1,111	254	375
9	**1,230**	0	0
Total	5,149	**1,726**	**23,467**

consecutive symbols increases, i.e. it will be more rare to find an eight symbol substring (e.g. "29476831"), than it would be to find a two symbol substring (e.g. "82").

The third component of the algorithm, the mismatch matrix (subsection 3.7.2.4), has a fixed space requirement of $O(n * l)$.

4.2.2 Performance.

As stated in subsection 3.7.1, only the daily segmentation of the dataset was processed with this method. Tables 4.2 to 4.6 show all relevant details of the execution of all five trajectory-derived symbol concatenations from subsection 3.6.1.4, and both versions of the log-likelihood ranking matrix (asymmetric and symmetric, subsection 2.4.1.2).

From an analysis of the data contained in the tables, four facts are observed in direct relation to the increase of the keyword length:

1. The number of potential candidate segments decreases.

2. The number of keywords searched increases, with certain boundaries.

3. The number of exact overlapping matches found increases, with boundaries as well.

4. The execution time increases.

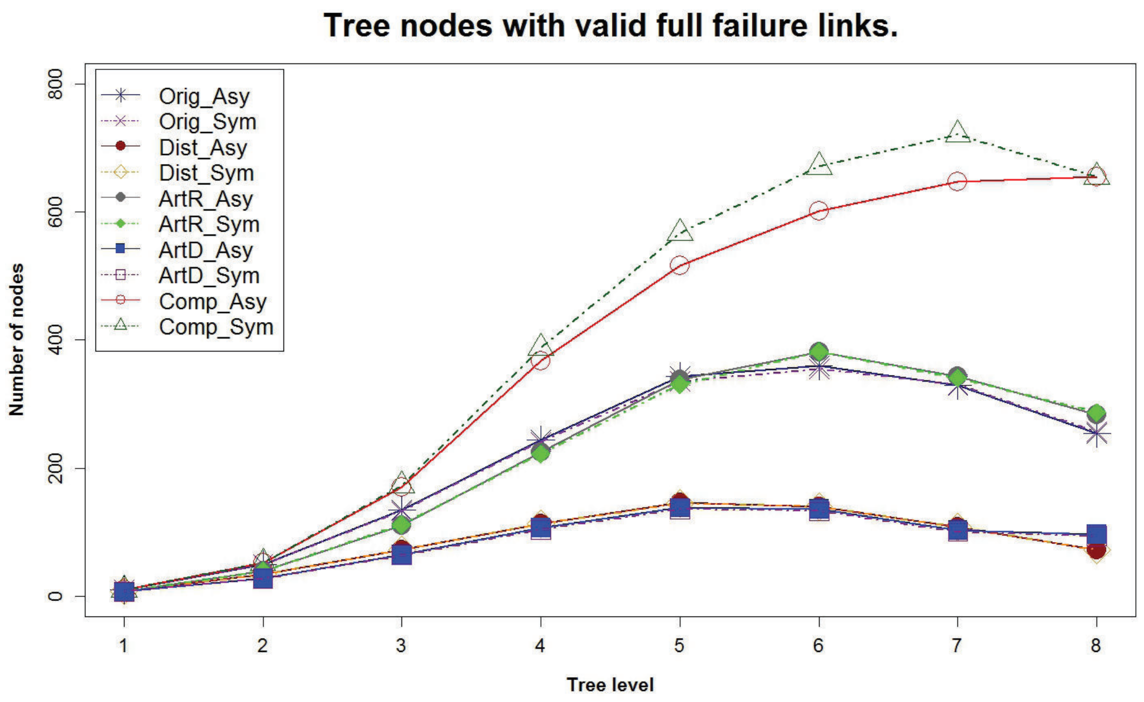

Figure 4.1: Tree nodes with valid full failure links by tree type and level, n = 9.

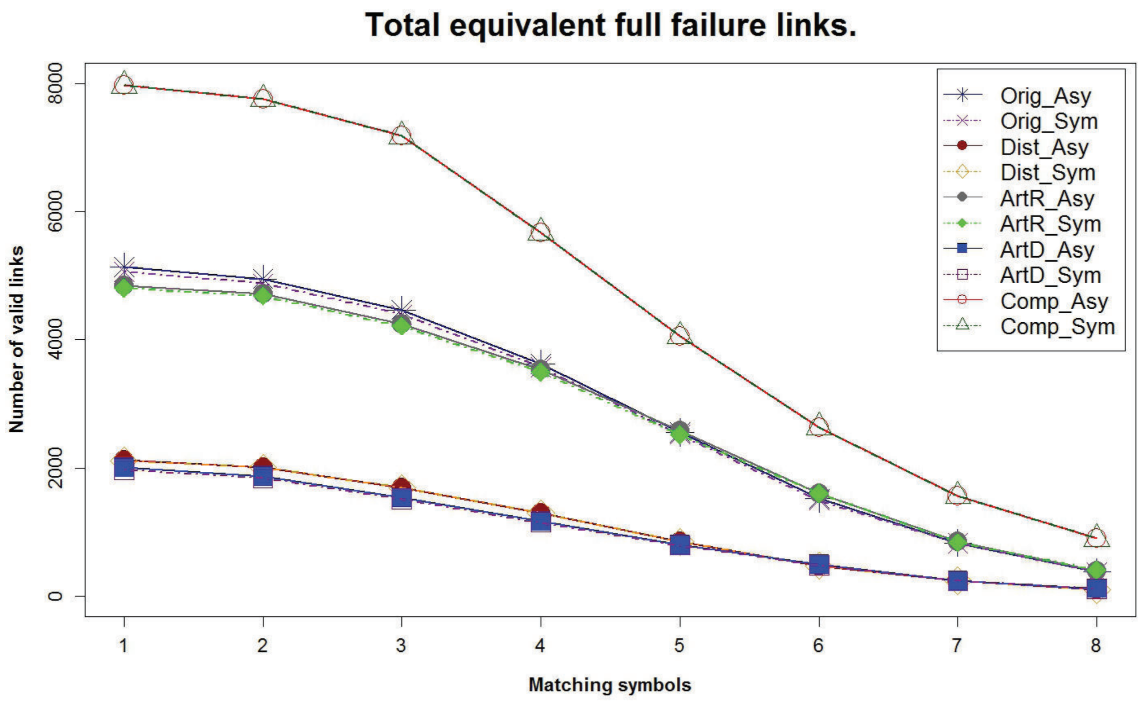

Figure 4.2: Equivalent full failure links by tree type and number of matching symbols, n = 9.

Table 4.2: Performance, original symbols.

Keyword length	Available segments	Asymmetric: Keywords searched	Asymmetric: Matches found	Asymmetric: Execution time (sec)	Symmetric: Keywords searched	Symmetric: Matches found	Symmetric: Execution time (sec)
3	17,808	90	2,446	54	92	2,453	56
6	7,801	749	24,778	89	762	25,439	90
9	5,508	1,375	16,076	89	1,376	16,008	91
12	4,520	1,312	9,622	132	1,317	9,340	129
15	3,688	1,192	6,011	134	1,173	5,952	136
20	2,174	797	2,909	117	804	2,920	124
25	1,022	418	1,737	154	418	1,734	165
30	996	385	1,380	127	387	1,382	121
40	967	231	825	123	232	826	137

Table 4.3: Performance, distinct symbols.

Keyword length	Available segments	Asymmetric: Keywords searched	Asymmetric: Matches found	Asymmetric: Execution time (sec)	Symmetric: Keywords searched	Symmetric: Matches found	Symmetric: Execution time (sec)
3	10,577	70	739	11	72	757	10
6	3,329	360	2,650	29	355	2,576	28
9	1,782	444	1,090	11	451	1,103	12
12	969	268	375	18	267	374	17
15	673	172	192	11	172	192	11
20	406	101	108	4	103	110	4
25	182	46	46	1	46	46	1
30	47	12	12	0.05	12	12	0.05
40	0	0	0	0	0	0	0

Table 4.4: Performance, artificial repeated symbols.

Keyword length	Available segments	Asymmetric: Keywords searched	Asymmetric: Matches found	Asymmetric: Execution time (sec)	Symmetric: Keywords searched	Symmetric: Matches found	Symmetric: Execution time (sec)
3	17,692	75	2,801	49	75	2,806	48
6	7,760	689	24,732	77	700	25,386	79
9	5,496	1,287	16,072	108	1,298	16,003	106
12	4,510	1,295	9,661	120	1,263	9,305	116
15	3,683	1,198	6,013	129	1,163	5,901	126
20	2,163	800	3,037	138	803	3,023	136
25	1,021	414	1,786	126	409	1,768	133
30	996	390	1,398	243	392	1,379	241
40	967	232	829	249	233	830	271

Table 4.5: Performance, artificial distinct symbols.

		Asymmetric			Symmetric		
Keyword length	Available segments	Keywords searched	Matches found	Execution time (sec)	Keywords searched	Matches found	Execution time (sec)
3	10,511	60	1,623	11	57	1,585	11
6	3,304	356	3,448	25	360	3,555	24
9	1,776	436	1,060	10	445	1,073	11
12	967	269	352	19	276	360	9
15	673	171	193	11	173	195	11
20	406	101	105	4	101	105	4
25	180	46	46	1	45	45	1
30	47	12	12	0.05	12	12	0.05
40	0	0	0	0	0	0	0

Table 4.6: Performance, completed symbols.

		Asymmetric			Symmetric		
Keyword length	Available segments	Keywords searched	Matches found	Execution time (sec)	Keywords searched	Matches found	Execution time (sec)
3	23,336	65	1,248,143	282	64	1,248,142	282
6	23,336	612	1,220,162	502	614	1,220,156	496
9	23,336	1,313	1,274,290	747	1,303	1,273,739	753
12	23,336	1,871	1,197,705	780	1,902	1,197,708	796
15	23,336	2,100	1,124,637	856	2,106	1,124,733	861
20	23,336	2,662	973,450	1304	2,698	973,568	1306
25	23,336	2,472	835,831	1509	2,472	835,831	1518
30	23,336	3,140	686,923	2148	3,164	687,196	2061
40	23,336	2,447	377,290	2228	2,446	377,289	2246

The decrease in potential candidate segments is valid for all tree types, except the "completed" variations. This can be explained because, in this segmentation every segment is forced to have exactly 72 symbols, corresponding to a regular division of a 24 hour period in 20 minute intervals (see subsection 3.6.1.4). This tendency is better observed in figure 4.3.

The bounded increase in the number of keywords searched can be explained by the empirical limitations of the dataset. In other words, not all possible subsequences occur in the dataset (e.g. "01234567890123456789" can't be found in the tracking dataset). Figure 4.4 illustrates this characteristic. Please note the sharp decrease in this metric at 9 symbols per keyword for all tree variations except the "completed" variants. This

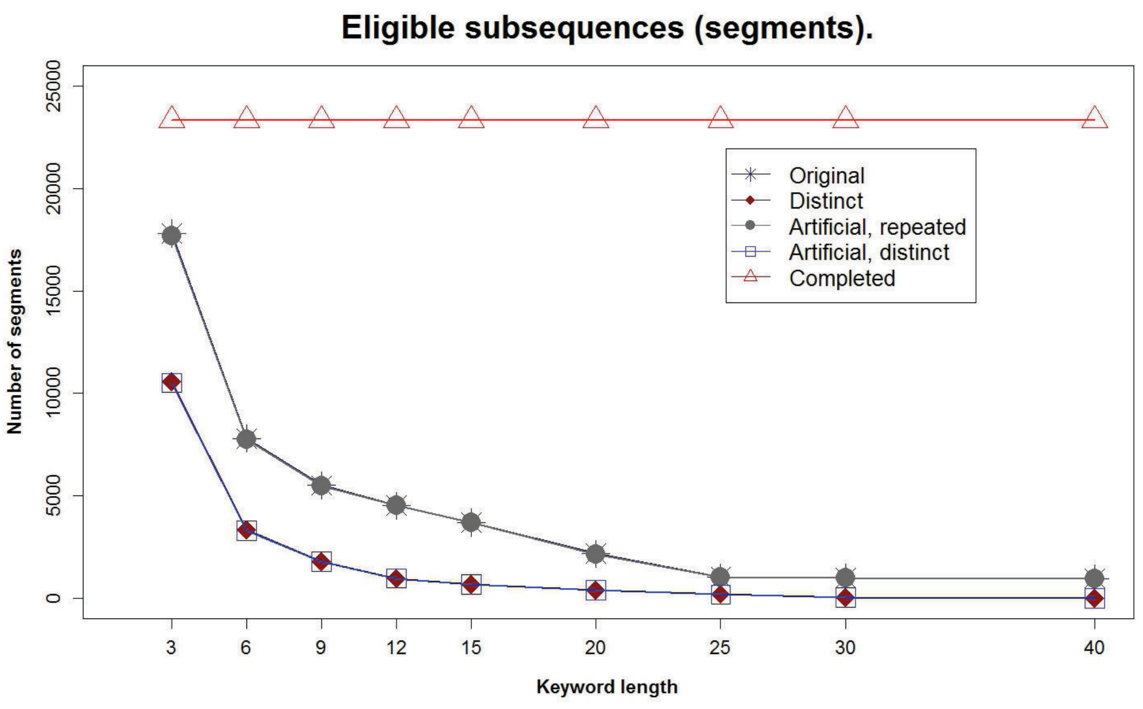

Figure 4.3: Number of eligible text subsequences (segments) by tree type and keyword size.

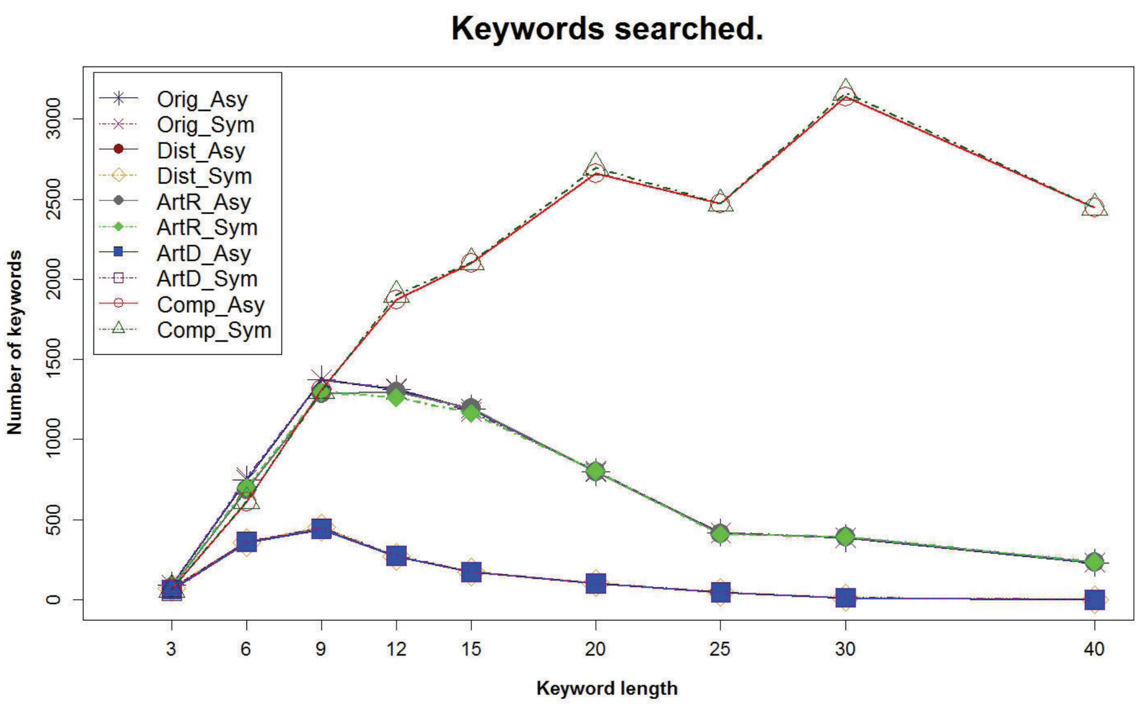

Figure 4.4: Number of keywords searched by tree type and keyword size.

behavior suggests that the higher concentration of information in the dataset exists for subsequences between 6 and 12 symbols in length. Further statistical tests can be conducted to confirm or refute this observation, but this is also out of the scope for this project.

To better explain the bounded increase in the number of exact overlapping matches found, its results have been divided into two illustrations. Figure 4.5 contains the graph for all variations except the “completed” trees. In this figure, there is a perceptible inflection point for keywords of 6 symbols in length, that can be explained by the combination of the bounded candidate segment decrease and the bounded searched keywords increase described above. The case of the “completed” trees, figure 4.6, can be seen as the empirical worst case scenario of the factors just described. Note that this data also contains an inflection point for keywords of 6 symbols in length, however, its direction is opposite from those of all the other tree types.

The execution time increase in figure 4.7 shows a predominantly linear behavior by tree type, even for the “completed” versions, the major difference amongst these linear tendencies being the slope of the fitted straight lines (not included in the figure). This empirical result agrees with previous analysis of the Boyer-Moore and the Aho-Corasick methods, including all variations and improvements used in this work [1, 23, 47, 48, 69, 123, 124, 137, 147, 148, 207, 231] (see subsection 2.4.1.1).

4.2.3 Performance improvement.

Once the comprehensive executions were finished, it was determined that additional Database Administrator (DBA) techniques could be used to improve the performance of the method. Keywords with 9 symbols were selected as a case study for this purpose.

Since the specific DBA techniques used for this part of the work are not of a program-

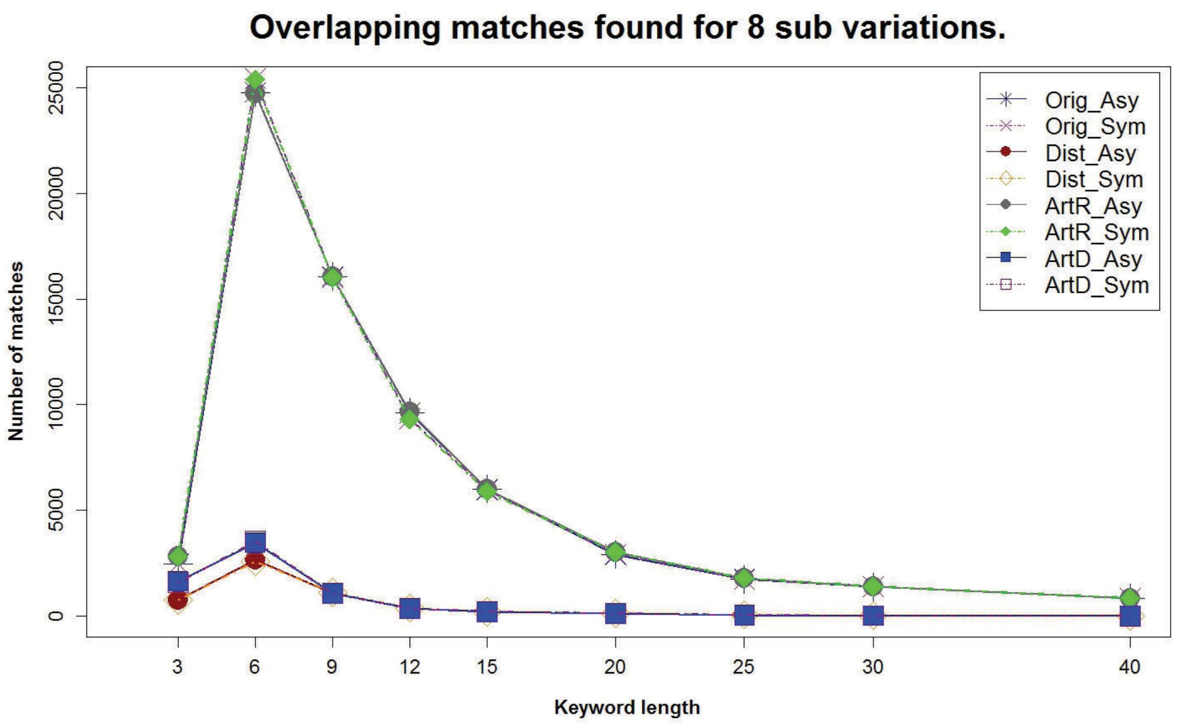

Figure 4.5: Number of overlapping matches found by tree type and keyword size.

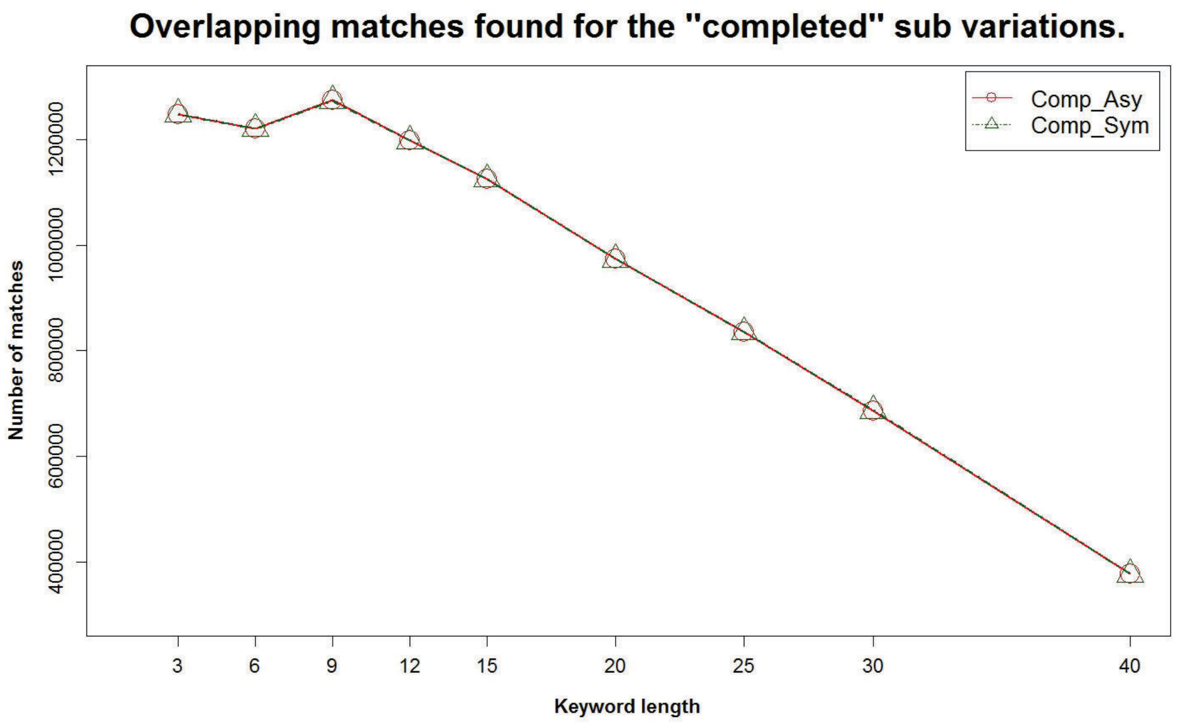

Figure 4.6: Number of overlapping matches found by tree type and keyword size (cont.).

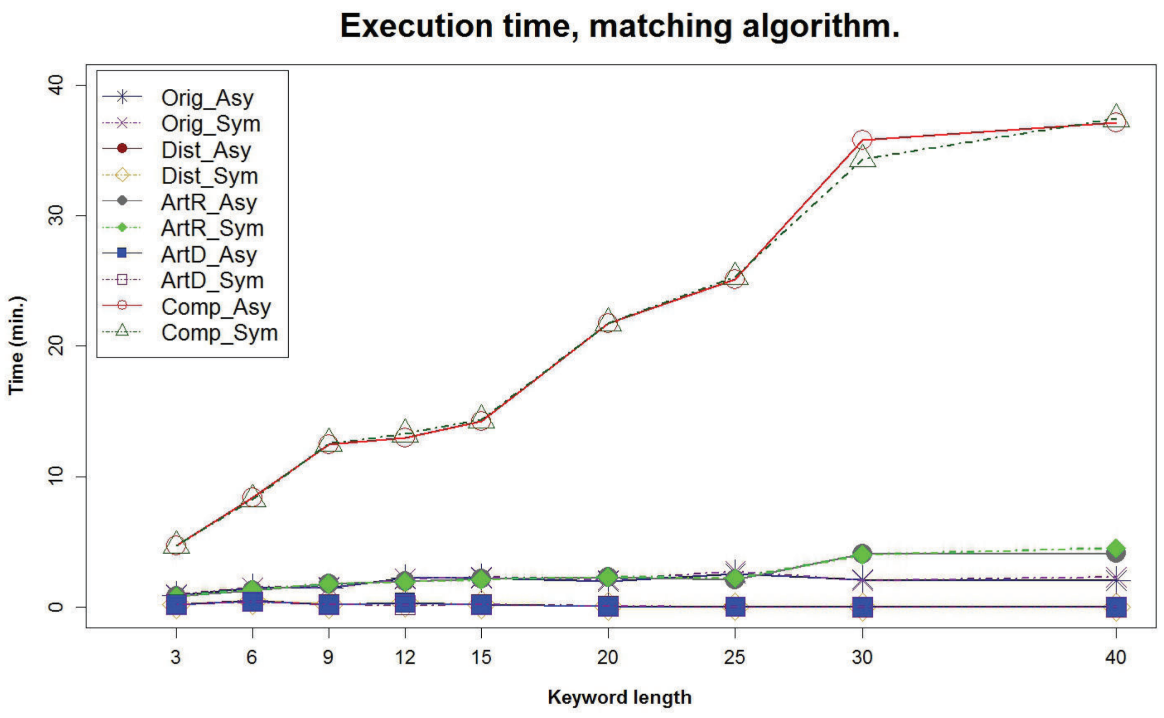

Figure 4.7: Execution times for the matching algorithm (in minutes) by tree type and keyword size.

ming nature, and will be different for almost every benchmark platform, their details are considered to fall outside the scope of this thesis and will not be presented. However, these administration techniques include complementary indexes [104], pinning [244], materialized views [244], on-the-fly views [104], pagination [244], temporary tablespace to table block synchronization [244], partitions [104], subpartitions [104], etc.

For each tree type, table 4.7 shows the total amount of individual character matches to perform in the worst case scenario. The values in column α correspond to the maximum number of available segments per keyword length (column “Available segments” in tables 4.2 to 4.6), which are constant for the cleaned dataset.

The number of characters in table 4.7 (column m) were measured directly from the database at the time of each algorithm execution. These values are lower than the total number of individual GPS positions for the cleaned dataset (1999,223, see section C.1),

Table 4.7: Number of individual elements to process, n = 9.

	Available segments			Keywords		Total elements	Total characters
Main variation	α	(characters) m	Sub variation	β	$o = \beta * n$	$\alpha * \beta$	$m * o$
Original	5,508	136,256	Asymmetric	1,230	11,070	6,774,840	1,508,353,920
			Symmetric	1,218	10,962	6,708,744	1,493,638,272
Distinct	1,782	26,278	Asymmetric	466	4,194	830,412	110,209,932
			Symmetric	465	4,185	828,630	109,973,430
Artificial, repeated	5,496	135,999	Asymmetric	1,207	10,863	6,633,672	1,477,357,137
			Symmetric	1,195	10,755	6,567,720	1,462,669,245
Artificial, distinct	1,776	26,184	Asymmetric	462	4,158	820,512	108,873,072
			Symmetric	454	4,086	806,304	106,987,824
Completed	23,336	1,680,192	Asymmetric	2,134	19,206	49,799,024	32,269,767,552
			Symmetric	2,134	19,206	49,799,024	32,269,767,552

in all cases except for the “completed” tree types, because only circadian segmentations with a minimum of 9 symbols were considered, and that quantity has been shown to diminish with the increase in keyword length (subsection 4.2.2 and figure 4.3).

The values in column β in table 4.7 include the average number of subsequences matched by the ten experiment executions. These values are the equivalent to column “Keywords searched” in tables 4.2 to 4.6 for keywords of length 9, but are numerically different as a consequence of the random nature of the candidate keyword sampling.

Table 4.8 shows the execution times of the matching algorithm (in seconds) before and after additional DBA optimizations. The execution times before optimizations were taken from the corresponding cells in tables 4.2 to 4.6.

Only the “distinct asymmetric” sub variation does not register any percentage change in table 4.8. The rest of the sub variations register a negative percentage change (“artificial, distinct asymmetric” and “artificial, distinct symmetric”), or a positive percentage change (all other seven sub variations).

The sub variations that show the smallest percentage changes in table 4.8 (the four “distinct” sub variations, i.e. “distinct asymmetric”, “distinct symmetric”, “artificial, distinct asymmetric” and “artificial, distinct symmetric”) correspond to the sub variations with the smallest amounts of total characters to process in table 4.7. The opposite is true of the sub variations that show the largest execution improvements, i.e. the sub

Table 4.8: Execution times for the matching algorithm (before and after further improvements), n = 9.

		Execution time		Percentage
Main variation	Sub variation	Before	After	change
Original	Asymmetric	89	61	31.46
	Symmetric	91	63	30.77
Distinct	Asymmetric	11	11	0.00
	Symmetric	12	11	8.33
Artificial, repeated	Asymmetric	108	67	37.96
	Symmetric	106	64	39.62
Artificial, distinct	Asymmetric	10	12	-20.00
	Symmetric	11	12	-9.09
Completed	Asymmetric	747	414	44.58
	Symmetric	753	416	44.75

variations in table 4.8 with the largest performance improvements correspond to the sub variations in table 4.7 with the largest amounts of total characters to process.

Clarifying in absolute execution time terms, the worst change in the performance of the algorithm after DBA modifications, the "artificial, distinct asymmetric" sub variation, only adds 2 seconds to the total execution time of the process. In contrast, the best performance change of the algorithm, the "completed symmetric" sub variation, reduces the entire processing time by 5 minutes and 37 seconds per execution.

All performance changes in table 4.8 are consistent with the tendency of the Boyer-Moore algorithm to perform worst for shorter keyword/text lengths, and better for larger keyword/text lengths [124, 137]. Therefore, it is expected that the algorithm/database incorporating DBA fine tunning techniques shall execute faster as compared to the version without DBA modifications.

4.2.4 Most frequent patterns.

The results from all variations are different, particularly between main variations, but all results show some of the same tendencies. For a brief detailed analysis, only the results from the "original asymmetric" variation will be presented here (tables 4.10 to 4.13), appendix D contains the equivalent tables for the other nine tree sub variations.

For this explanation, the term *n-repetition* will be used to denote all patterns of any size which are composed entirely of consecutive repetitions of the symbol "n", e.g. 4-repetition denotes "44", "444", "4444", "44444", "444444", etc. With the same purpose, the symbol "x" means "any number and configuration of symbols" before or after its occurrence, e.g. x4x denotes "0**4**", "3572**4**765", "555555555**4**7777777", etc.

To make this analysis easier, the equivalences between the abstract symbols of the method and the land covers classes of the original digital map (figure 3.5.b, and table 3.2) are reproduced in table 4.9.

Table 4.9: Equivalents, land cover symbols to land cover classes.

Symbol	Land cover
0	"n/a"
1	Upland trees
2	Wetland trees
3	Upland herb
4	Wetland herb
5	Shrub
6	Water
7	Barren
8	Snow / Ice
9	Cloud
A	Shadow

The following general tendencies can be observed in tables 4.10 to 4.13:

1. All pattern frequencies (support) decrease as n increases.
2. At $n = 3$, the most unique patterns take place.
3. The 5-repetition pattern (Shrub) is the most frequent of the n-repetition patterns, except at $n = 3$.

4. The n-repetition patterns maintain the same frequency order from $n = 6$ to $n = 20$ (inclusive).

5. At $n = 25$, the 2-repetition pattern (Wetland trees) disappears.

6. At $n = 30$, the 3-repetition (Upland herb) and the 7-repetition (Barren) patterns swap order.

7. Ignoring the n-repetition patterns, the patterns x51x (Shrub-Upland trees), x35x (Upland herb-Shrub), x71x (Barren-Upland trees), and their symmetrics, occur with the highest frequency.

The same tendencies can be observed in the tables presented in appendix D, with the exception of the tendencies specifically identified for n-repetition patterns that do not appear for the "distinct" variations. This is correct, and an independent validation of the segmentation rules presented in subsection 3.6.1.4.

The first observation of the list, the pattern frequency decrease as n increases, can be explained by the fact that it is statistically more likely to find shorter subsequences as compared to a longer subsequences [78, 157]. In other words, it is more likely to find the pattern "555551", than it is to find the pattern "5555555555555555555555555551", etc. See subsection 4.2.1 for a related discussion under empirical conditions on tree density and full failure links.

Figure 4.8 shows the histogram of the most frequent patterns at $n = 3$, which is consistent in general tendency for all values of n in all the results of this work (tables 4.10 to 4.13, and D.1 to D.32). Please note that almost all cases show one order of magnitude decrease in frequency (i.e. support) between the most frequent pattern and the second most frequent pattern.

To help explain the rest of the observations of the list, starting with the second, the most unique patterns occurring at $n = 3$, let us consider the relative abundance of each

Table 4.10: Original asymmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
222	1,042	555555	7,016	555555555	4,185	555555555555	2,887
444	345	333333	2,695	333333333	1,388	333333333333	779
000	165	777777	2,132	777777777	1,070	777777777777	593
141	86	222222	440	555555551	292	555555555551	135
666	59	333335	270	155555555	270	222222222222	125
161	53	533333	231	222222222	227	155555555555	113
244	52	777771	226	333333335	135	555555555511	86
442	47	333331	224	533333333	106	115555555555	84
144	46	177777	223	555555557	94	333333333335	77
441	45	111777	216	444444444	91	555555155555	63
422	36	133333	205	777777771	86	533333333333	47
224	34	111333	200	177777777	84	555555555557	45
414	26	333111	199	111117777	75	777777777771	39
142	25	117777	183	333333353	75	177777777777	37
1A1	23	333311	183	755555555	75	333333333355	37
241	16	113333	174	133333333	73	755555555555	35

Table 4.11: Original asymmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
555555555555555	2,161	55555555555555555555	1,456
333333333333333	435	33333333333333333333	167
777777777777777	340	77777777777777777777	143
222222222222222	72	22222222222222222222	33
555555555555551	72	15555555555555555555	25
155555555555555	57	55555555555555555511	23
333333333333335	49	55555555555555555553	21
555555555555553	44	35555555555555555555	17
355555555555555	41	11155555555555555555	14
115555555555555	37	53333333333333333333	11
555555555555511	36	55555555555555155555	10

Table 4.12: Original asymmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
5555555555555555555555555	1,099	555555555555555555555555555555	873
3333333333333333333333333	86	777777777777777777777777777777	52
7777777777777777777777777	74	333333333333333333333333333333	47
5555555555555555555555511	13		
5555555555555555555555553	11		

Table 4.13: Original asymmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
5555555555555555555555555555555555555555	559
7777777777777777777777777777777777777777	33

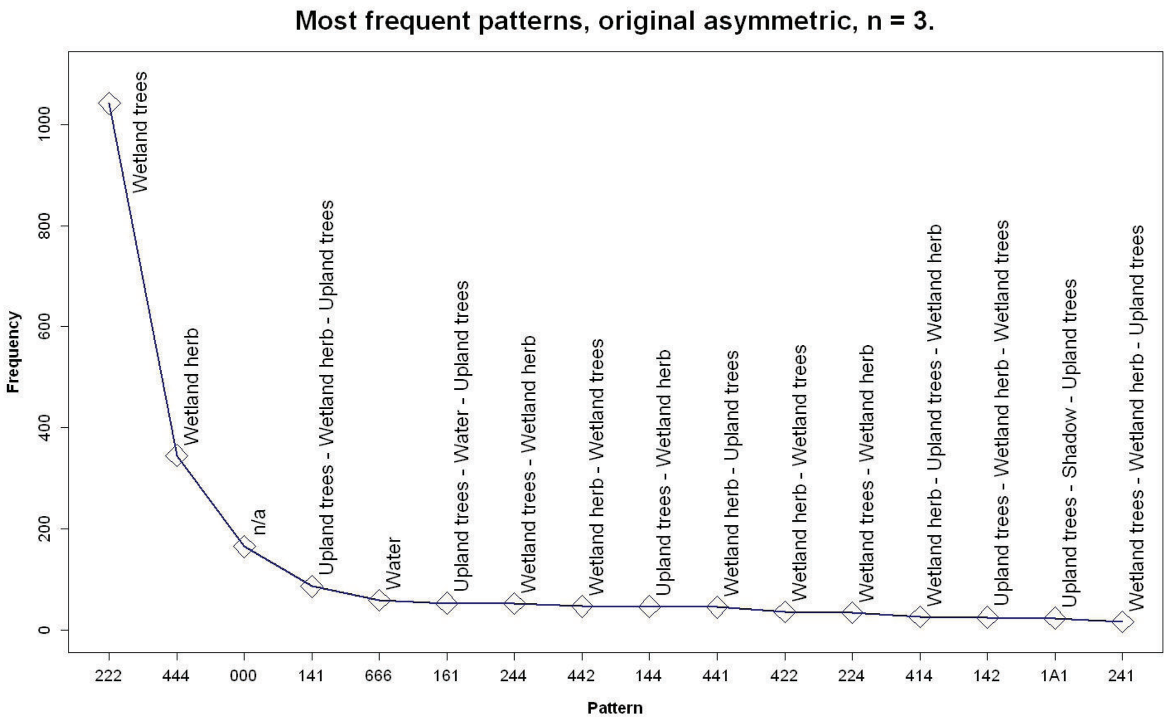

Figure 4.8: Most frequent patterns, original asymmetric, n = 3.

symbol in the context of the full cleaned dataset. To do so, the percentage data from table 3.2 (section 3.4) is presented in table 4.14 in individual symbol frequency terms for all segmentations (subsection 3.6.1.4), and with the explicit addition of the “n/a” class.

The uniqueness of the patterns at $n = 3$ can be explained by the relative abundance of their individual symbols as the size of the searched patterns increase, i.e. observing that symbol “4” in table 4.14 (“Wetland herb”), column “Original”, is the sixth most abundant symbol with only 1,072 occurrences in total, it is to be expected that its individual relevance would be seriously attenuated as n increases. In other words, it is statistically expected that the frequencies of relatively rare symbols measured with a

Table 4.14: Individual symbol frequencies by segmentation type.

Symbol	Land cover	Original	Distinct	Artificial, repeated	Artificial, distinct	Completed
0	“n/a”	484	183	0	0	1,481,933
1	Upland trees	105,793	33,514	105,793	33,482	105,793
2	Wetland trees	3,736	2,060	3,736	2,060	3,736
3	Upland herb	21,693	11,389	21,693	11,385	21,693
4	Wetland herb	1,072	571	1,072	571	1,072
5	Shrub	44,975	20,638	44,975	20,625	44,975
6	Water	325	222	325	216	325
7	Barren	20,673	11,534	20,673	11,525	20,673
8	Snow / Ice	163	130	163	130	163
9	Cloud	6	6	0	0	6
A	Shadow	303	250	0	0	303
	Total	199,223	80,497	198,430	79,994	1,680,672

log-likelihood function would be more relevant at high sensitivity levels (e.g. n =3), as compared to their prominence at low sensitivity levels (e.g. n =25) [10, 78, 157].

This is particularly evident in table 4.10 at $n = 3$ in the cases of the 6-repetition pattern and patterns “161” and “1A1”, that disappear from the rest of the tables of most frequent patterns. Their unique symbols, “6” (“Water”), and “A” (“Shadow”) occupying places eighth and ninth in table 4.14. The 0-repetition pattern (“n/a”) is a perfect example of this phenomenon because its individual symbol occupies the seventh place in the list of most frequent symbols, and it only appears as frequent at $n = 3$. The rest of the most frequent patterns at this sensitivity level are composed by combinations of symbols “1” (“Upland trees”), “2” (“Wetland trees”), and “4” (“Wetland herb”), which are expected to lose statistical relevance as n increases, as stated above.

One last explanation about this observation constitutes the reappearance of the 4-repetition pattern at $n = 9$ in table 4.10, but nowhere else. Considering that symbol “4” occupies the sixth place in the list of most abundant individual symbols (table 4.14), this could be due to Tobler’s first law of Geography [239].

The third observation of the previous list, the 5-repetition pattern (“Shrub”) being the most frequent pattern at n =[6, ... 40], points to the effectiveness of the log-likelihood ranking method (subsection 2.4.1.2) to compensate for under-represented symbols (classes) in the data. Symbol "5", "Shrub" terrain in the land cover map of the study area (figure 3.5.b, and table 3.2), is the fourth most abundant class (table 4.14). The unique patterns at $n = 3$ are additional indications of this feature of the method.

With the same logic, the method demonstrates its effectiveness to compensate for over-represented symbols by excluding the 1-repetition pattern ("Upland trees") from the lists of results at all sensitivity levels. In other words, the 1-repetition pattern is trivial or uninteresting for this analysis.

An additional effect of the properties of the log-likelihood ranking matrix is the frequency proximity of patterns such as "333335" and their symmetric "533333", meaning that, for the log-likelihood ranking matrix it appears to be the same to present the pattern forwards or backwards judging by the contents of tables 4.10 to 4.13, and D.1 to D.32.

Observations 4 to 6 of the previous list can be explained by the combined effect of the relative abundance of each individual symbol, the mathematical expression of the log-likelihood ranking matrix, and the first law of Geography [239], as n increases (i.e. as the level of sensitivity of the method decreases). All other position changes of equivalent patterns can also be explained by this combination of factors. For example, the downwards "movement" of pattern "777771", originally at eleventh place at $n = 6$ to thirteenth place at $n = 12$.

The high frequency of patterns x51x, x35x, x71x, and their symmetrics can also be explained by the combined effect of the first law of Geography [239], the relative abundance of symbols "1" ("Upland trees"), "5" ("Shrub"), "3" ("Upland herb"), and "7" ("Barren"), first to fourth most abundant symbols respectively, and the compensations introduced by the log-likelihood ranking matrix for under and over-represented symbols (subsection 2.4.1.2).

4.2.5 Behavioral interpretation of some frequent patterns.

The prevalence of patterns containing a high quantity of one symbol, followed by a few occurrences of another symbol, and their symmetrics, seems to indicate behavior patterns identified in previous studies (e.g [114, 183, 46]). For example, the pattern "333333333335" (Upland herb - Shrub, table 4.10 at $n = 12$), appears to reflect the preference of grizzly bears to spend considerable amounts of time in areas of high canopy cover, followed by short excursions to areas of lower canopy vegetation [24, 109, 217].

Other frequent patterns, such as "555555555551" (Shrub - Upland trees, table 4.10 at $n = 12$) indicate what could be foraging behaviors [70, 130, 188]. Additionally, similar patterns involving relatively extended periods of barren terrain use, followed by the use of high canopy areas, for example "777777777771" (Barren - Upland trees, table 4.10 at $n = 12$), appear to indicate travel and/or foraging behaviors [52, 139].

These three examples agree with previous studies that have identified the preference of grizzly bears to use the edges of highly vegetated areas for the majority of their activities, with occasional visits to lower canopy vegetation areas and barren terrain [39, 40, 58, 166, 173, 189, 190].

To make further distinctions between these patterns, the addition and analysis of time intervals between consecutive GPS positions should offer more detailed information. Seasonal food availability should help explain some small differences between similar behaviors at different times of the year, for example, foraging between upland trees and upland herbs, as compared to foraging between upland herbs and shrub.

4.2.6 Graphic representation of some frequent patterns.

As an example of the results of the clustering step in the method (section 3.8), figure 4.9 shows three patterns detected in the trajectories for the example bear of section 3.3

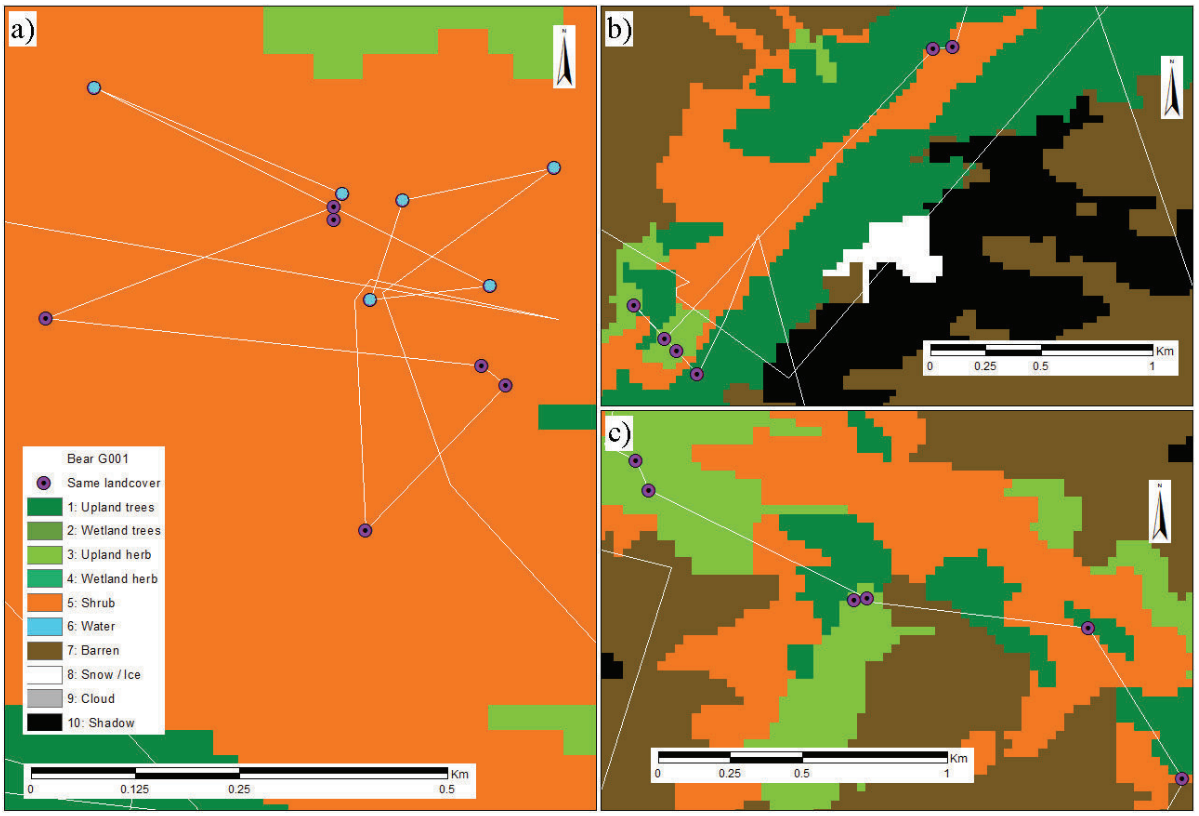

Figure 4.9: Most frequent patterns identified for bear G001 at keyword length = 6, a) 555555 (Shrub) twice consecutive, b) 113331 (Upland trees - Upland herb - Upland trees), and c) 333355 (Upland herb - Shrub)

(G001). Please note the different scales at which each sub figure is presented.

Figure 4.9.a shows two consecutive occurrences of pattern “555555” (Shrub), that was detected with the clustering method defined in [221], the so called “multiple non-overlapping subsequences in the same diagonal” (see section 3.9). Figure 4.9.b shows pattern “113331” (Upland trees - Upland herb - Upland trees), which was detected with DBSCAN [128], and could be interpreted as a foraging behavior (see previous subsection), and figure 4.9.c (also detected with DBSCAN), shows pattern “333355” (Upland herb - Shrub), which could be a different expression of foraging, probably due to seasonal changes (see previous subsection as well).

Technically speaking, these sub figures serve as examples of the effectiveness of the, "multiple non-overlapping hits in the same diagonal segment" step in the method developed by Altschul et al. [7, 9, 10] (subsections 2.4.1.2, and 3.8). The detection of the tendency of grizzly bears to predominantly utilize the edges of relatively small vegetated patches, which was previously detected in past studies (e.g. [166, 189]), can be attributed to the visual exploration of the bidimensional representation of the results of the current work, and further supports the decision of sections 3.8 and 3.9 to evaluate and present the results of this work in map form.

4.2.7 Comparison with other methods.

To compare the results and performance of this method with other solutions, fully integrated or not, there are two immediate approaches. Process the cleaned grizzly bear dataset from this work with the applications from the previous solutions, or take the datasets from those previous approaches and process them with the current solution.

As far as this work is aware, there is no source code freely available from previous methods that accomplish a similar task as the current implementation, or their objectives are not to detect frequent movement patterns. In practical terms, these dissimilarities would require to either re-tool the previous implementations in a fully integrated solution with altered objectives (to detect frequent movement patterns), or changing the current application to fit the objectives of the previous solutions. In either case, each alternative represents several months of additional programming work that are not available for the current project.

Instead, the best performance of the current solution is given in table 4.15 for keywords of nine symbols in length (i.e. $n = 9$). These quantities were obtained by dividing the total number of characters to process in the worst case scenario (table 4.7, column "Total characters") by the execution times after DBA improvements (table 4.8, column

Table 4.15: Best performance of the current solution, n = 9.

Main variation	Sub variation	Best performance (MOPS)
Original	Asymmetric	24.73
	Symmetric	23.71
Distinct	Asymmetric	10.02
	Symmetric	10.00
Artificial, repeated	Asymmetric	22.05
	Symmetric	22.85
Artificial, distinct	Asymmetric	9.07
	Symmetric	8.92
Completed	Asymmetric	77.95
	Symmetric	77.57

"Execution time after"). These performances are given in Millions of Operations Per Second (MOPS).

4.3 Chapter summary.

This chapter presented extensive results and analysis of the local alignment approach to, from the point of view of the current work, solidify its suitability as an effectively analysis tool for the studied grizzly bear tracking dataset. Included are its main and secondary memory space requirements, and its performance profile, including its sensitivity response.

The results from this method were presented in tabular and graphical form with a list of their main characteristics, explanations of their main tendencies, and analysis of some of their most apparent exceptions. The presented results, their analysis, and some of its major implications were identified as agreeing with some previous ecological and behavioral studies of grizzly bears.

Chapter 5

Conclusions

The following conclusions can be directly derived from the implementation of all methods, unsuitable and suitable alike, as described in the previous chapters and complementary appendixes:

• Avoid dynamic programming to achieve linear-time processing algorithms.

In general terms, dynamic programming methods have a worst case execution time of $O(n^2)$, while other methods have lower execution times. [1, 23, 47, 48, 69, 123, 124, 137, 138, 147, 148, 182, 187, 207, 209, 210, 228, 231, 235]

• Eliminate trivial patterns from the dataset as early as possible[*].

The reduction of the size of the dataset to process, directly translates into shorter execution times. [5, 6, 7, 8, 9, 10, 28, 37, 77, 122, 131, 142, 143, 167, 168, 201, 205, 219, 221, 245, 248, 250]

• Hierarchical search trees have demonstrated to be good alternatives to organize collections of patterns for exact matching searches.

Since the mid '60s, many data structures used to organize large collections of data have been evaluated. Over the years, hierarchical tree structures have proven to be the fastest and most versatile. [2, 3, 4, 29, 30, 31, 32, 33, 34, 35, 41, 49, 53, 56, 63, 64, 65, 66, 67, 71, 99, 113, 119, 121, 125, 133, 144, 156, 170, 171, 172, 191, 200, 202, 220]

[*]This will mostly depend on the analyst's level of experience.

• From the point of view of this work, full RDBMS integration has demonstrated its viability.

The RDBMS integration has shown that implementing complicated data structures and manipulation algorithms does not have to be at the cost of diminished performance. See sections 3.7.1 and 4.2.

An additional advantage of a fully integrated solution is the availability of advanced DBA techniques to increase performance. These techniques are not available to external libraries, and severely limited to solutions that employ add-on RDBMS libraries [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. See subsection 4.2.3.

5.1 Answers to the research questions.

1. What are the environmental factors that have a greater influence on grizzly bear activities?

Human activity, **seasonal food availability**, and **competition**, amongst others. See sections 2.1, 3.2 and 3.4 for further details.

2. Is there a machine learning method that can do all of the following at the same time?

Local alignment.

Inclusion and/or exclusion of specific patterns can be conducted in a convenient manner, without requiring programming modifications to the analysis tools. Since the method was implemented as an incremental process, including and excluding specific subsequences of interest is a trivial matter. The researcher only needs to make a pause after the set of candidate subsequences has been generated, insert or delete the subsequences of special interest, and let the process continue from that point with no need for additional programming changes.

The scalability and sensitivity of the method has been established with repeated execution of the same operational code with variations on the segmentation, keyword length and ranking method. See section 4.2.

The inclusion of multicriteria similarity evaluations is easily accomplished by changing the definition and construction of the log-likelihood ranking matrices. See subsection 2.4.1.2.

The RAM and secondary storage space requirements of the method were established under empirical conditions and are of modest demands as compared to its theoretical worst case scenario. See subsection 4.2.1.

In a similar manner, the execution times of the matching algorithm were established under several empirical conditions, including its empirical worst case scenario. Under such conditions, the execution of the method is expected to follow a linear behavior. See subsections 4.2.2 and 4.2.3.

See sections 2.4.1.1, 3.7.1 and 4.2 for additional detailed descriptions of the operational parts and results of the local alignment method implementation.

3. Can a modern RDBMS support this type of analysis tools (i.e. efficient and scalable) using only internal resources?

Yes. The successful fully integrated implementation of all three machine learning methods (sections 2.4.1.1, 3.7.1 and 3.7.2, chapter 4, and appendix A), regardless of their suitability status, helps support the robustness and maturity of RDBMSs as a viable solution platform. In particular, the suitability of the local alignment method has shown that a modern (i.e. fully featured) RDBMS provides the necessary programming elements to fully support a complex integrated solution of this type, without necessarily having to reduce its performance.

The adoption of the minimalistic approach simplified some of the usual programming decisions (e.g. in the case of the local alignment method, not to use recursion), reducing at the same time the spectrum of applicable programming techniques, without severely limiting the availability of performance enhancement techniques [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. See subsections 2.2, 3.7.1 and 4.2 for additional details.

4. Can the method be explained in graphical form to make it more accessible to other developers and researchers?

Yes. If the reader is not familiar with the use of hierarchical tree structures for exact matching purposes, subsections 2.4.1 and 3.7.1 should serve as a general introduction to the task at hand. Otherwise, subsection 3.7.2 on its own, presented an extensive graphical representation of all the details and programming decisions necessary to arrive at an efficient implementation of this method.

5. Is there a topological analysis theory capable of explaining all aspects of moving object patterns based on fundamental movement characteristics (e.g. velocity, direction of travel)?

No. The proposed frameworks are either solutions to special cases that restrict their validity to a small portion of similar cases, or theoretical starting points that include generic cases with very restricted conditions. In both cases, the proposed frameworks are works in progress that do not offer comprehensive coverage of all possible moving object cases. See appendix B for extensive additional details on these frameworks.

5.2 Contributions.

The contributions of this work originate mostly from the implementation of the suitable method, the local alignment approach (sections 2.4.1.1, 3.7.1, 3.7.2, and 4.2):

- Automatic detection of rare movement patterns at different sensitivity levels across temporal resolutions.

Based on the over and under-representation of land cover classes in the set of GPS positions (section 3.4, table 3.2), and the set of frequent patterns found by the local alignment analysis tool implemented (subsection 4.2.4, and appendix D), it is the opinion of this work that the log-likelihood ranking method (subsection 2.4.1.2, equation 2.1) has demonstrated its effectiveness in highlighting rare movement patterns at different sensitivity levels (i.e with sets of keywords of different length), without the need for separate analysis of data acquired at different temporal resolutions.

- Fully integrated implementation of methods inside an RDBMS.

Regardless of their suitability for the grizzly bear dataset, in the opinion of this work, the success in the implementation of its three methods, helps support the notion that RDBMSs have achieved a level of robustness and flexibility that will allow them to efficiently support any fully integrated complex analysis tool (i.e. any complex data structure and programming method).

For the RKT implementation in particular, as previously discussed, the minimalistic implementation philosophy adopted for the current work reduced the programming techniques available for this solution, particularly the selection of abstract RAM data structures [13, 14, 15, 16, 17, 18, 19, 20, 21, 22].

To make the application's source code adhere to these demands, the empirical portion of the work determined that it was best not to utilize complex RAM data structures for the storage or manipulation of the tracking dataset. Instead, relatively simple relational tables, combined with relatively complicated procedural manipulation techniques, were employed to solve the needs of the process, including the full implementation of RKTs. See sections 3.7.1 and 3.7.2 for additional details.

- Graphical representation of the exact matching process of a keyword set.

The modified graphical representation developed in this work to illustrate the exact matching processes, and the abstract data structures required to conduct a simultaneous search of a set of keywords (of any length), against a relatively longer text, enriches the available didactic tools to explain these advanced concepts.

To overcome some of the conceptual difficulties of understanding the counterintuitive nature of the Boyer-Moore method [47, 48], that analyzes the text from left-to-right and the pattern from right-to-left, combined with the multitude of simultaneous demands of matching an entire set of keyword subsequences from the Aho-Corasick method [1], subsection 3.7.2 presented a step-by-step graphical representation of all the special cases, multiple conditions that a combined implementation requires, and some of the viable solutions to simultaneously satisfy all needs.

- Comprehensive performance analysis of the exact matching process.

The repeated execution of the entire process with different combinations of keyword length, circadian segmentation, and log-likelihood ranking method, allowed the construction of detailed tabular summaries of the behavior of the local alignment method (see section 4.2). The empirical results of these experimentations strongly suggest that the empirical worst case for RAM and secondary storage space requirements are much lower than the upper bound of their theoretical equivalents (subsections 4.2.1 and 4.2.2).

The strongly linear behavior of the execution times of the method (figure 4.7), considering the volume of text segments available (figure 4.3), and the quantity of keywords to search for at each keyword length (figure 4.4), indicate a predictable response in execution as the volume of processed data increases, i.e. a reasonably scalable response.

The analysis of the most frequent patterns at different keyword lengths, illustrates the level of sensitivity of the overall process. The integrated compensation mechanism, the formulation of the log-likelihood matrices, demonstrates its efficiency to minimize the relevance of over-represented classes at all sensitivity levels, as well as its ability to highlight the uniqueness of under-represented classes at high levels of sensitivity. See subsection 4.2.4 for additional explanations.

- Modified use of Forwardtrack.

As far as this work is aware, forwardtrack has not been used in this way before, or in a fully integrated solution, as it was done for the current implementation.

The empirical concept of forwardtrack, based on the knowledge of "the next character from the current tree node", helps speed up the process in three different places in the algorithm. At the construction time of the RKT, it helps minimize the set of equivalent full failure links for every node in the tree (algorithm 3.2). After a mismatch, it helps minimize the set of candidate nodes for "branch jumping" (algorithm 3.3), and once an *n-repetition* subsequence has been detected in the text, it efficiently registers all other subsequent overlapping *n-repetitions* (algorithm 3.4). See subsections 3.7.2.5 and 4.2.2 for additional details.

- Modified use of Backtrack.

To the best knowledge of this work, the concept of backtrack tree traversal has not been implemented before as part of the tree itself, or in a fully integrated method, as presented in this document.

The need to preserve the maximum number of matches after a mismatch, combined with the use of hierarchical tree data structures, requires tree traversal (downwards and upwards). Two alternatives are immediately available for this, recursion and position-aware traversal with the use of a stack. Both methods are available inside an RDBMS [13, 14, 15, 16, 17, 18, 19, 20, 21, 22], but the use of recursion incorporates additional unpredictable factors for a concurrent system, such as the amount of memory needed for successive recursive calls for each user, the switching speed between user environments (RAM image switching), pagination, buffering, etc.

To supply the backwards traversal functionality, avoiding the additional complications of using recursion or an independent stack structure, the current work utilized a fully incorporated backtrack mechanism that combines the information already available in the tree structure with purpose-specific markers (i.e. additional columns in the corresponding relational tables). In essence, solving these needs with a relatively complex iterative approach. See algorithm 3.7 for the particulars.

From the previous supporting results, I believe that the research objectives have been met.

5.3 Future work.

In the judgment of this work, the included areas of future research in this section are considered of special interest because, they will either offer important complementary empirical characteristics of the local alignment method, or have the potential to improve its performance even further. The major limiting factor that forced this work to leave these alternatives pending, was their additional development time, estimated to vary from a few weeks, to more than six months in each case.

Without changes to the existing code, additional sensitivity tests for all values of n

from 2 to 11 (inclusive), can be conducted to establish an even more detailed sensitivity response profile for the local alignment method, than the one already included in this work. Approximately three weeks of work should be required to execute all the necessary processes for this task.

Generalization to collections of patterns with different number of characters per keyword is supported by the same processing algorithms, requiring minimum programming changes to the implementation. One week of modifications, and one additional week of unit testing are judged to be enough to implement this enhancement. Between four and five weeks of additional processing should be required to confirm the correctness and advantages of this generalization for all the scenarios presented in section 4.2.

The 20 minutes temporal coverage concept from subsection 3.6.1.4, reflecting the time interval frequency distribution of the dataset (subsection 3.3.1), proved to be an effective method of minimizing the subsequence sizes to analyze, maintaining at the same time all the topological relationships of the original data. However, other temporal coverage definitions are possible that still maintain the characteristic of being an integer sub multiple of all the time intervals with high frequency in figure 3.4, including the 20 minutes interval, e.g. 10 minutes, 5 minutes, 2 minutes. Additional development time in the range of five to seven months, would be required to implement these alternatives to determine the relative advantages and disadvantages of each one of them.

Significant additional development time, initially estimated in the range of seven to nine months, prevented a full exploration of an integrated multicriteria technique that would combine the log-likelihood ranking method with several other data sources such as gender, elevation, distance to roads (as a surrogate of the intensity of human activity), grizzly bear density maps (as a surrogate of competition), etc.

Similarly, additional development time in the order of three to six months, would have been needed to test all the similarity formulas, single and multicriteria alike, combined with full analysis of GPS points at individual temporal resolutions (i.e. 4 hrs, 2 hrs, 1 hr and 20 minutes, plus any other implemented temporal coverage intervals).

Additional improving concepts, academic and practical, such as the full development of the Reorganized Reversed Keyword Trees (RRKTs) and Override Comparison Vectors (OCVs), as briefly described in the following subsections, can be applied to the current implementation in order to increase its flexibility and/or performance. The RRKT and OCV concepts were initially explored in this work, but suspended because of the already mentioned lack of development time. The details of the achieved exploration can be found in the following subsection (5.3.1).

Implementation of an end-user interface would require from four to seven months, but would transform the current solution into a user friendly application more attractive to other research groups. Making it possible for other experts in the field to easily evaluate, and possibly improve the current code, when applied to other datasets and objectives.

5.3.1 Reorganized reversed keyword trees (RRKTs).

To improve the performance of the original string matching problem, Sunday [235] preprocessed the text to produce a simple table of individual symbol probabilities that can be used to generate a scanning order vector for the pattern to search for. The purpose of this vector is to match the rarest character in the pattern with the corresponding initial position of the text, in order to find a mismatch as early in the process as possible, and to slide the pattern accordingly. In the other hand, if a match is found, the process continues with the second rarest character in the pattern, and so on. With each successive character match, the overall probability of a successful match increases as well.

This principle can easily be applied to any set of keyword subsequences to generate a single scanning order vector, provided two conditions. First, whatever the selected evaluation criteria, it has to be applied uniformly to all keyword subsequences. Second, the resulting scanning order vector will be used to build a new keyword tree, a Reorganized Reversed Keyword Tree (RRKT), to make the matching of the entire keyword set the most efficient process possible.

5.3.1.1 RRKTs for the design subsequences.

As presented since figure 3.13, the set of design subsequences (reversed or not) can be conceptualized as a sorted set of contiguous rows aligned by the position of their individual symbols (i.e. a two-dimensional matrix). As a consequence, any evaluation criteria can be applied with two direction components, horizontal and vertical. When the criteria lacks the vertical component, the evaluation can be said to be purely horizontal (i.e. by full subsequence). Conversely, when the criteria does not have a horizontal component, the evaluation operates exclusively within each tree level (i.e. by column).

There are as many evaluation criteria available as specific application objectives. Purely based on probabilities and the overall speed of the evaluation itself, the following alternatives were studied:

1. Subsequences by rarest symbol and tree node closer to the root. Figure 5.1.
2. Subsequences by full subsequence probability. Figure 5.2.
3. Tree levels by combined probabilities of symbols contained in each level. Figure 5.3.
4. Tree levels by rarest symbol. Figure 5.4.
5. Tree levels by most common symbol. Figure 5.5.
6. Tree levels by hierarchical individual probabilities from most common to rarest. Figure 5.6.

Figures 5.1 to 5.6 show each one of the RRKTs resulting by the application of the criteria described above (respectively).

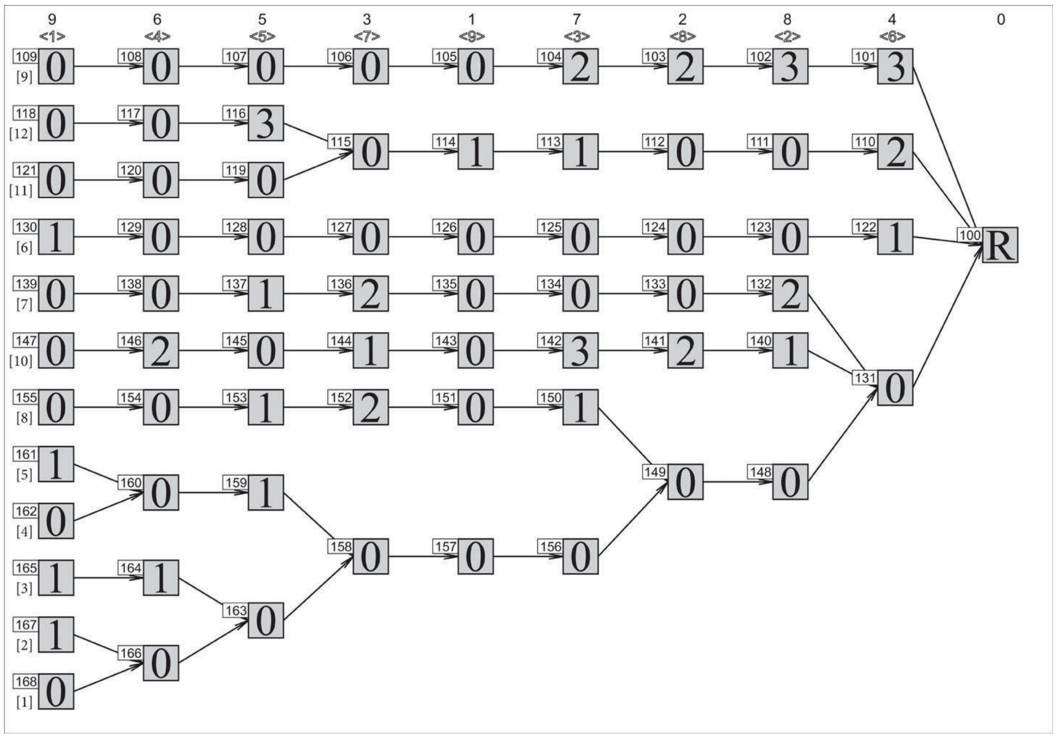

Figure 5.1: Reorganized reversed keyword tree with levels sorted by rarest symbol and closer node to the root.

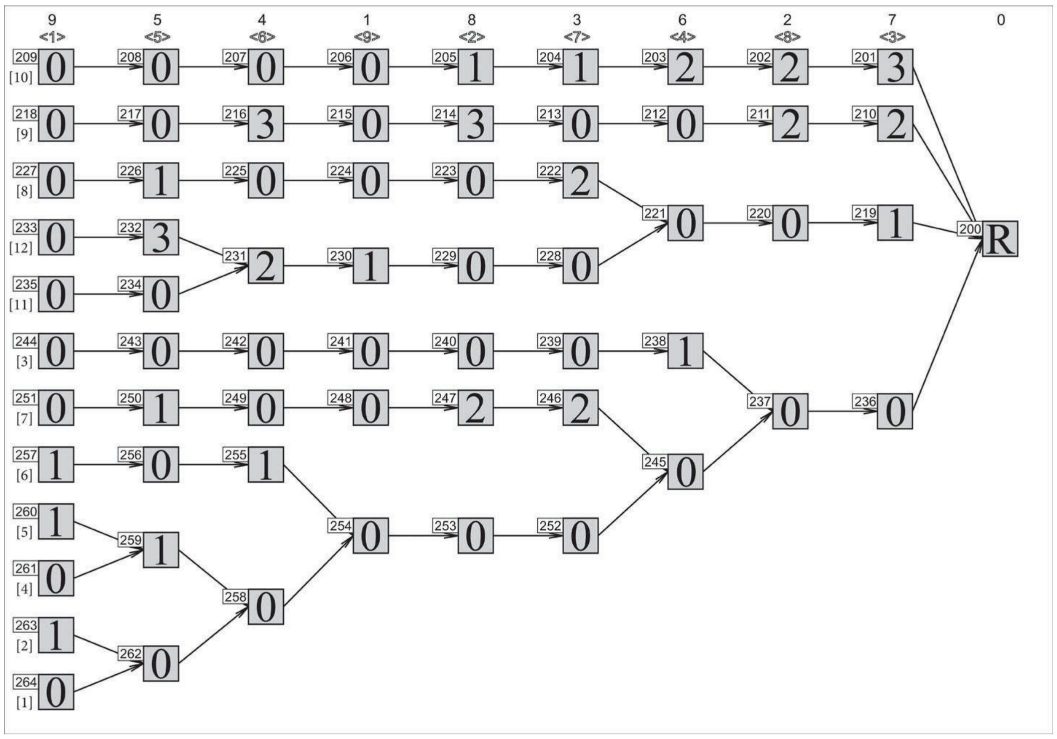

Figure 5.2: RRKT by full subsequence probability.

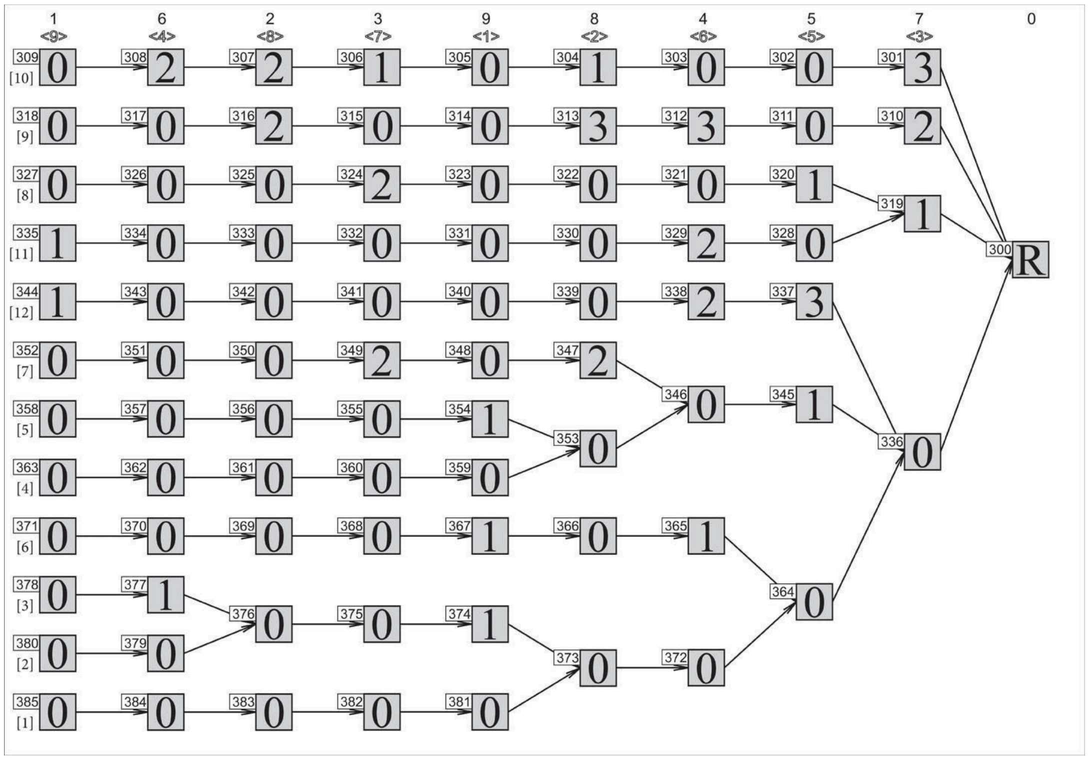

Figure 5.3: RRKT by tree level and combined probability of all symbols.

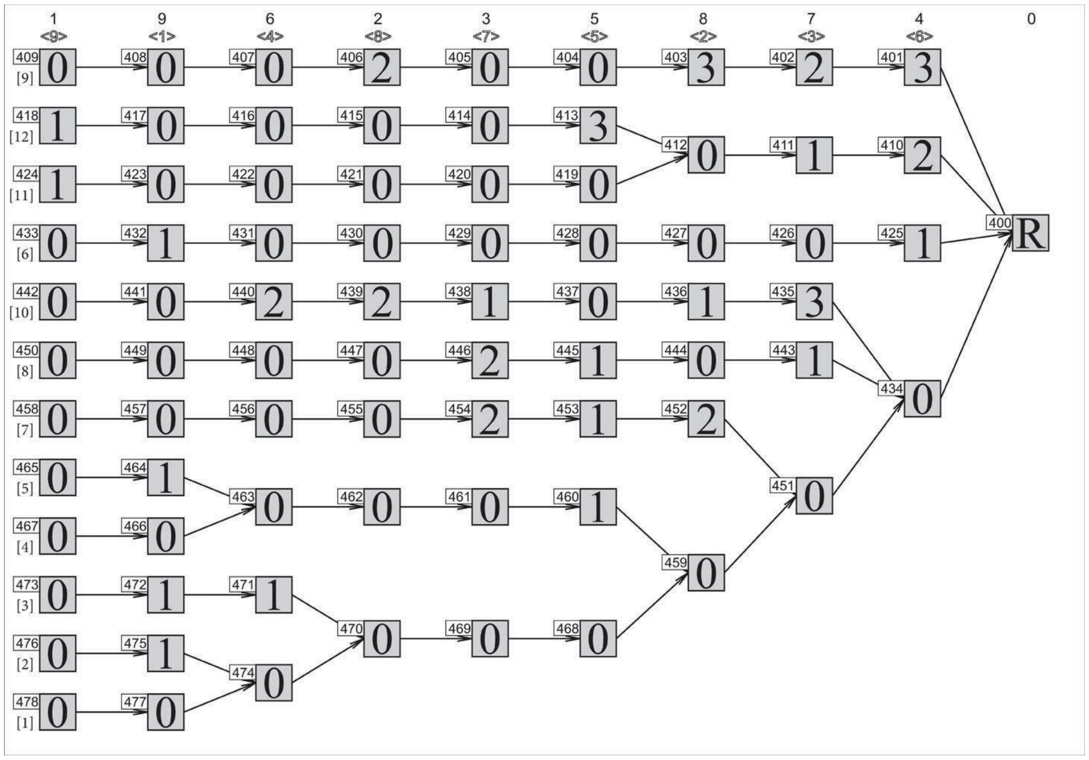

Figure 5.4: RRKT by tree level and probability of the individual rarest symbols.

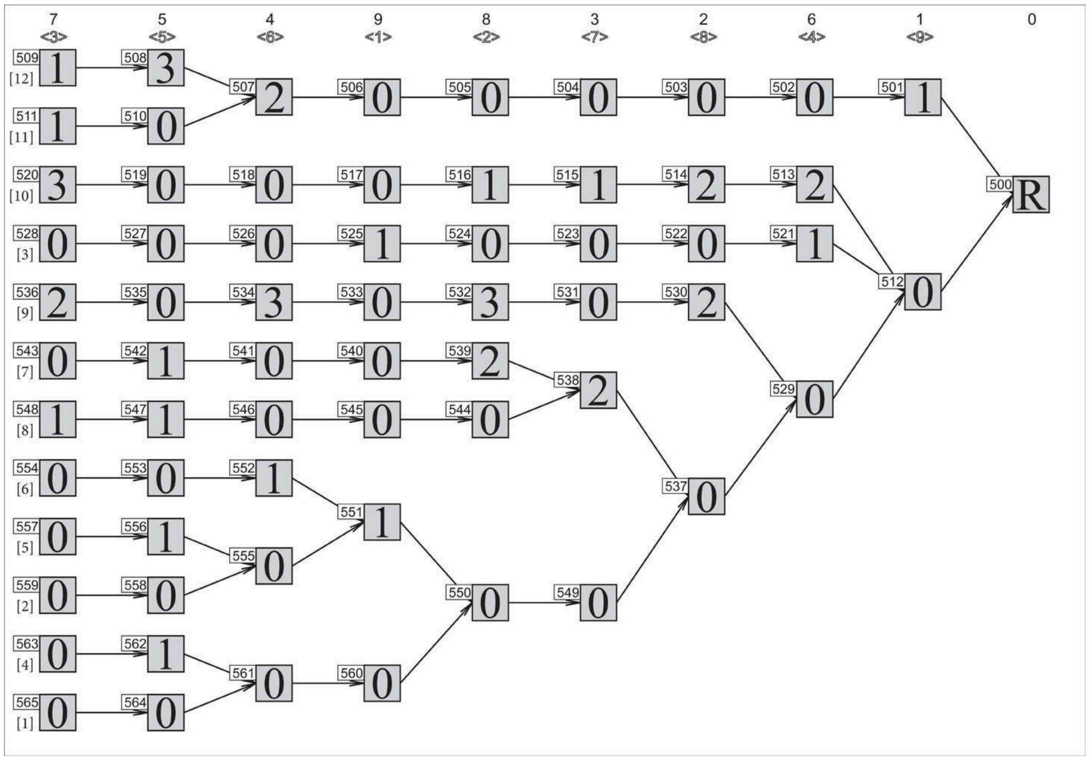

Figure 5.5: RRKT by tree level and the probability of the most common symbol.

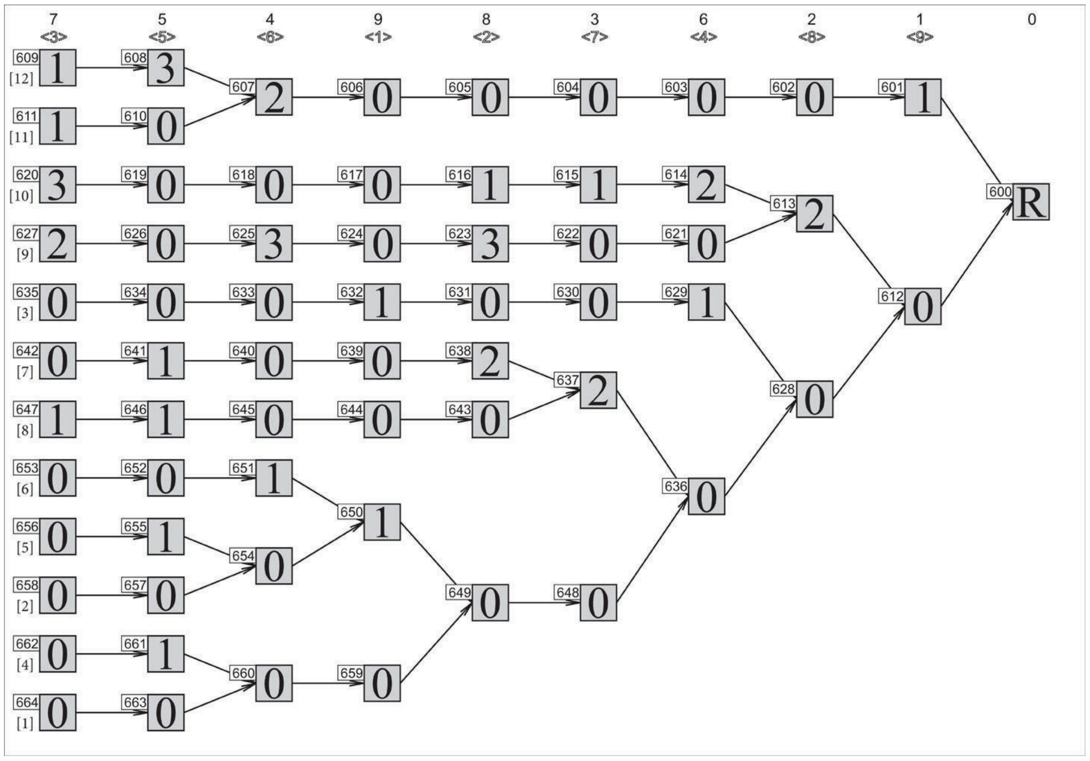

Figure 5.6: RRKT by tree level and symbols' probability (most common to rarest).

As far as this research is aware, there is no mathematical theory that provides an absolute best solution for this case. However, the objective remains to make the matching process as efficient as possible. Therefore, following the concept by Sunday, testing early for the rarest symbol in a single pattern, appears to be the best alternative.

The information from the comparison vector for each tree can be easily incorporated in the matching process. Table 5.1 shows the full set of Override Comparison Vectors (OCVs) for this case, in which, a one (1) value means "match as usual" (i.e. keep traversing the tree), while a zero (0) means "automatic match" (i.e. skip this tree level match).

Table 5.1: Override comparison vectors (sorted).

Order	Original	Reversed
a	000000000	000000000
b	000001000	000100000
c	010001000	000100010
d	010001010	010100010
e	011001010	010100110
f	011001011	110100110
g	011001111	111100110
h	011011111	111110110
i	011111111	111111110
j	111111111	111111111

5.3.1.2 Failure links for RRKTs.

This development continued until a prototype algorithm for the calculation of the full failure links for the RRKTs was achieved, but the lack of overall time forced these efforts to be abandoned at this stage. Figure 5.7 shows the first RRKT presented in this section (figure 5.1) with each node including its corresponding accumulated subsequence, while figure 5.8 shows the best full failure links found for this case.

The calculation of the full failure links for this type of trees involves the evaluation

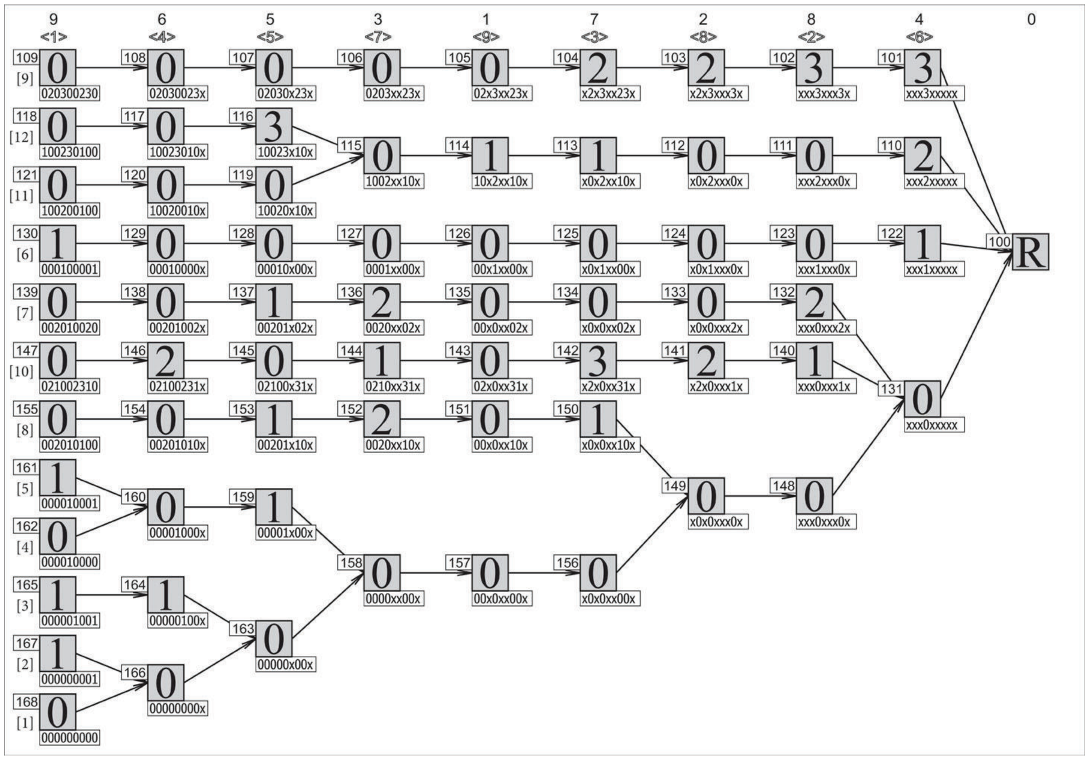

Figure 5.7: Cumulative subsequences for the first variation of the RRKT (figure 5.1).

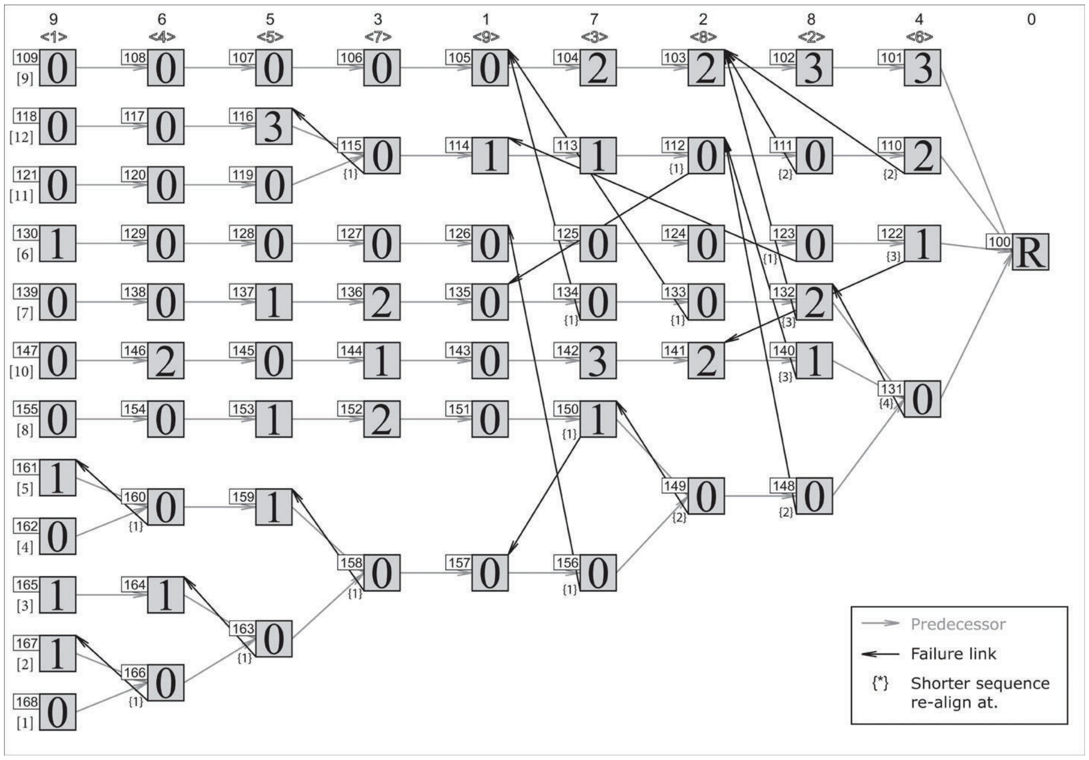

Figure 5.8: Full failure links for the first variation of the RRKT (figure 5.1).

of all possible alignments of the partial subsequences with all potential target nodes. This implies that this calculation can quickly become a combinatorial problem with exponential geometric growth. This is the case of the achieved prototype calculation algorithm in the current work, and an additional area of potential improvement.

For such development, this work recommends the use of the forwardtrack principle (subsection 3.7.2.5) as a first refinement method to reduce the potential target set of each node.

Bibliography

[1] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, June 1975.

[2] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Addison Wesley, October 2007.

[3] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation, and Compiling I: Parsing*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1972.

[4] ——, *The Theory of Parsing, Translation, and Compiling II: Compiling*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1974.

[5] S. F. Altschul, "Amino acid substitution matrices from an information theoretic perspective," *Journal of Molecular Biology*, vol. 219, no. 3, pp. 555–565, June 1991.

[6] ——, "BLAST algorithm," in Encyclopedia of Life Sciences, John Wiley & Sons, Inc. http://onlinelibrary.wiley.com/book/10.1002/047001590X, September 2005.

[7] S. F. Altschul, E. M. Gertz, R. Agarwala, A. A. Schffer, and Y.-K. Yu, "PSI-BLAST pseudocounts and the minimum description length principle," *Nucleic Acids Research*, vol. 37, no. 3, pp. 815–824, February 2009.

[8] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, October 1990.

[9] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein

database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[10] S. F. Altschul, J. C. Wootton, E. Zaslavsky, and Y.-K. Yu, "The construction and use of log-odds substitution scores for multiple sequence alignment," *Computational Biology*, vol. 6, no. 7, July 2010.

[11] M. Andersson, J. Gudmundsson, P. Laube, and T. Wolle, "Reporting leaders and followers among trajectories of moving point objects," *Geoinformatica*, vol. 12, no. 4, pp. 497–528, 2008.

[12] N. Andrienko and G. Andrienko, "Designing visual analytics methods for massive collections of movement data," *Cartographica*, vol. 42, no. 2, pp. 117–138, 2007.

[13] ANSI/ISO/IEC 9075-01, "Information technology-Database languages-SQL-Part 1: Framework (SQL/Framework)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-1, 2008.

[14] ANSI/ISO/IEC 9075-02, "Information technology-Database languages-SQL-Part 2: Foundation (SQL/Foundation)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-2, 2010.

[15] ANSI/ISO/IEC 9075-03, "Information technology-Database languages-SQL-Part 3: Call-Level Interface (SQL/CLI)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-3, 2008.

[16] ANSI/ISO/IEC 9075-04, "Information technology-Database languages-SQL-Part 4: Persistent Stored Modules (SQL/PSM)," American National Standards

Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-4, 2010.

[17] ANSI/ISO/IEC 9075-05, "Information technology-Database languages-SQL-Part 5: Host Language Bindings (SQL/Bindings)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-5, 1999.

[18] ANSI/ISO/IEC 9075-09, "Information technology-Database languages-SQL-Part 9: Management of External Data (SQL/MED)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-9, 2010.

[19] ANSI/ISO/IEC 9075-10, "Information technology-Database languages-SQL-Part 10: Object Language Bindings (SQL/OLB)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-10, 2010.

[20] ANSI/ISO/IEC 9075-11, "Information technology-Database languages-SQL-Part 11: Information and Definition Schemas (SQL/Schemata)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-11, 2008.

[21] ANSI/ISO/IEC 9075-13, "Information technology-Database languages-SQL-Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-13, 2010.

[22] ANSI/ISO/IEC 9075-14, "Information technology-Database languages-SQL-Part 14: XML-Related Specifications (SQL/XML)," American National Standards Institute/International Organization for Standardization/International Electromechanical Commission, Tech. Rep. 9075-14, 2010.

[23] A. Apostolico and R. Giancarlo, "The Boyer-Moore-Galil string searching strategies revisited," *SIAM Journal on Computing*, vol. 15, no. 1, pp. 98–105, February 1986.

[24] C. D. Apps, B. N. McLellan, J. G. Woods, and M. F. Proctor, "Estimating Grizzly bear distribution and abundance relative to habitat and human influence," *The Journal of Wildlife Management*, vol. 68, no. 1, pp. 138–152, January 2004.

[25] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, 1st ed. Cambridge University Press, April 2009.

[26] A. C. Atkinson, "A segmented algorithm for simulated annealing," *Statistics and Computing*, vol. 2, no. 4, pp. 221–230, 1992.

[27] J. Banks and J. S. Carson, *Discrete-Event System Simulation*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1984.

[28] J. Basch, L. J. Guibas, and J. Hershberger, "Data structures for mobile data," in *Proceedings of the Eighth Annual Association for Computing Machinery Symposium on Discrete Algorithms (ACM-SIAM)*, SIGACT: ACM Special Interest Group on Algorithms and Computation Theory. Philadelphia, PA, USA: SIAM: Society for Industrial and Applied Mathematics, 1997, pp. 747–756.

[29] R. Bayer, "Binary B-Trees for virtual memory," in *1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control.* New York, NY, USA: Association for Computing Machinery (ACM), 1971, pp. 219–235.

[30] ——, “Symmetric binary B-Trees: Data structure and maintenance algorithms,” *Acta Informatica*, vol. 1, no. 4, pp. 290–306, 1972.

[31] R. Bayer and E. M. McCreight, “Organization and maintenance of large ordered indexes,” in *1970 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*. Houston, TX, USA: Association for Computing Machinery (ACM), November 1970, pp. 107–141.

[32] ——, “Organization and maintenance of large ordered indexes,” *Acta Informatica*, vol. 1, no. 3, pp. 173–189, 1972, Note: Journal reprint of the original workshop paper from 1970.

[33] R. Bayer and J. K. Metzger, “On encipherment of search trees and random access files,” *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 1, pp. 37–52, March 1976.

[34] R. Bayer and M. Schkolnick, “Concurrency of operations on B-Trees,” *Acta Informatica*, vol. 9, no. 1, pp. 9–21, 1977.

[35] R. Bayer and K. Unterauer, “Prefix B-Trees,” *ACM Transactions on Database Systems (TODS)*, vol. 2, no. 1, pp. 11–26, March 1977.

[36] Bazaar, “The R environment for statistical computing and graphics,” http://www.r-project.org/, January 2012.

[37] M. Beckstette, R. Homann, R. Giegerich, and S. Kurtz, “Fast index based algorithms and software for matching position specific scoring matrices,” *BMC Bioinformatics*, vol. 7, no. 389, August 2006.

[38] P. Beier and R. F. Noss, “Do habitat corridors provide connectivity ?” *Conservation Biology*, vol. 12, no. 6, pp. 1241–1252, December 1998.

[39] B. Benn, "Grizzly bear mortality in the central Rockies ecosystem, Canada," Master's thesis, University of Calgary, Calgary, Alberta, Canada, 1998.

[40] A. Berland, T. Nelson, G. B. Stenhouse, K. Graham, and J. Cranston, "The impact of landscape disturbance on Grizzly bear habitat use in the Foothills Model Forest, Alberta, Canada," *Forest Ecology and Management*, vol. 256, pp. 1875–1883, 2008.

[41] H. J. Berliner, "The B*-Tree search algorithm: A best-first proof procedure," *Artificial Intelligence*, vol. 12, no. 1, pp. 23–40, May 1979.

[42] R. Billen and Y. Kurata, "Refining topological relations between regions considering their shapes," *Geographic Information Science*, vol. 5266, pp. 20–37, 2008.

[43] E. K. Blum and A. V. Aho, *Computer Science: The hardware, software and heart of it*, 1st ed. Springer - Verlag, December 2011.

[44] M. Bouden, B. Moulin, and P. Gosselin, "The geosimulation of West Nile virus propagation: A multi-agent and climate sensitive tool for risk management in public health," World Wide Web, International Journal of Health Geographics 7:35 (doi: 10.1186/1476-072X-7-35) http://www.ij-healthgeographics.com/content/7/1/35, July 2008.

[45] F. Bousquet and C. Le Page, "Multi-agent simulations and ecosystem management: A review," *Ecological Modeling*, vol. 176, no. 3-4, pp. 313–332, September 2004.

[46] M. S. Boyce, P. R. Vernier, S. E. Nielsen, and F. K. A. Schmiegelow, "Evaluating resource selection functions," *Ecological Modeling*, vol. 157, no. 2-3, pp. 281–300, November 2002.

[47] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," Stanford Research Institute and Xerox Palo Alto Research Center, Menlo Park and Palo Alto, CA,

USA, Tech. Rep., December 1975.

[48] ——, “A fast string searching algorithm,” *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, October 1977.

[49] L. Boytsov, “Indexing methods for approximate dictionary searching: Comparative analysis,” *Journal of Experimental Algorithmics*, vol. 6, no. 1, pp. 1–91, May 2011.

[50] R. Bradbury, *Futures, predictions and other foolishness*, ser. Complexity and Ecosystem Management: The Theory and Practice of Multi-agent Systems. School of Human Evolution and Social Change, Arizona State University, 2002, pp. 48–62.

[51] A. W. Burks, *Essays on Cellular Automata*, 1st ed. Champaign, IL, USA: University of Illinois Press, 1970.

[52] B. L. Carra, “Spatial and spatial-temporal analysis of Grizzly bear movement patterns as related to underlying landscapes across multiple scales,” Ph.D. dissertation, Wilfrid Laurier University, Waterloo, Ontario, Canada, 2010.

[53] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, “Searching in metric spaces,” *Computing Surveys*, vol. 33, no. 3, pp. 273–321, September 2001.

[54] J. Chen, C. Li, Z. Li, and C. M. Gold, “Improving 9-intersection model by replacing the complement with Voronoi region,” *Geo-spatial Information Science*, vol. 3, no. 1, pp. 1–10, March 2000.

[55] ——, “A Voronoi-based 9-intersection model for spatial relations,” *International Journal of Geographical Information Science*, vol. 15, no. 3, pp. 201–220, 2001.

[56] M. Chen, X. D. Gao, and H. F. Li, “Parallel DBSCAN with priority R-Tree,” in *The Second IEEE International Conference on Information Management and Engineering (ICIME)*, 2010, pp. 508–511.

[57] C.-L. B. Chetkiewicz, "Conservation corridors for carnivores: Integrating pattern and process in the Canadian Rocky Mountains," Ph.D. dissertation, University of Alberta, Edmonton, Alberta, Canada, Fall 2008.

[58] L. M. Ciarniello, M. S. Boyce, D. C. Heard, and D. R. Seip, "Components of Grizzly bear habitat selection: Density, habitats, roads, and mortality risk," *Journal of Wildlife Management*, vol. 71, no. 5, pp. 1446–1457, 2007.

[59] E. Clementini and R. Billen, "Modeling and computing ternary projective relations between regions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 799–814, June 2006.

[60] E. Clementini and P. Di Felice, "Topological invariants for lines," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 1, pp. 38–54, January/February 1998.

[61] E. Clementini, P. Di Felice, and P. van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," *Lecture Notes in Computer Science*, vol. 692, pp. 277–295, 1993.

[62] E. Clementini, S. Skiadopoulos, R. Billen, and F. Tarquini, "A reasoning system of ternary projective relations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 2, pp. 161–178, February 2010.

[63] L. G. W. A. Cleophas, "Towards spare time: A new taxonomy and toolkit of keyword pattern matching algorithms," Master's thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, Netherlands, August 2003.

[64] ——, "Tree algorithms: Two taxonomies and a toolkit," Ph.D. dissertation, De-

partment of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, Netherlands, April 2008.

[65] L. G. W. A. Cleophas, B. W. Watson, and G. Zwaan, "A new taxonomy of sublinear right-to-left scanning keyword pattern matching algorithms," *Science of Computer Programming*, vol. 75, no. 11, pp. 1095–1112, November 2010.

[66] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, June 1970.

[67] ——, *The Relational Model for Database Management: Version 2*, 1st ed. Reading, MA, USA: Addison Wesley Publishing Company, April 1990.

[68] C. A. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, June 2004.

[69] R. Cole, "Tight bounds on the complexity of the Boyer-Moore string matching algorithm," in *Proceedings of the Second Annual Association for Computing Machinery Symposium on Discrete Algorithms (ACM-SIAM)*, SIGACT: ACM Special Interest Group on Algorithms and Computation Theory. Philadelphia, PA, USA: SIAM: Society for Industrial and Applied Mathematics, 1991, pp. 224–233.

[70] A. Collingwood, S. E. Franklin, X. Guo, and G. B. Stenhouse, "A medium-resolution remote sensing classification of agricultural areas in Alberta Grizzly bear habitat," *Canadian Journal of Remote Sensing*, vol. 35, no. 1, pp. 23–36, 2009.

[71] D. Comer, "The ubiquitous B-Tree," *Computing Surveys*, vol. 11, no. 2, pp. 121–137, June 1979.

[72] B. J. Copeland, *Alan Turing's Automatic Computing Engine: The master code-breaker's struggle to build the modern computer*, 1st ed. Oxford, NY, USA: Oxford University Press, April 2005.

[73] COSEWIC, *COSEWIC Assessment and Update Status Report on the Grizzly Bear Ursus Arctos in Canada.* Ottawa: Committee on the Status of Endangered Wildlife in Canada, 2002.

[74] J. J. Craighead, "Studying Grizzly habitat by satellite," *National Geographic Magazine*, vol. 150, no. 1, pp. 148–158, July 1976.

[75] J. J. Craighead, G. B. Scaggs, and J. S. Sumner, "A definitive system for analysis of Grizzly bear habitat and other wilderness resources," *Research Reports - National Geographic Society*, vol. 16, pp. 153–162, 1975.

[76] J. J. Craighead, J. S. Sumner, and G. B. Scaggs, *A definitive system for analysis of Grizzly bear habitat and other wilderness resources utilizing Landsat multispectral imagery and computer technology*, ser. Monograph no. 1. Missoula, MT, USA: Wildlife-Wildlands Institute, University of Montana, 1982.

[77] M. B. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, *A model of evolutionary change in proteins.* Washington, DC, USA: National Biomedical Research Foundation, 1978, pp. 345–352.

[78] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *A Modern Introduction to Probability and Statistics*, 1st ed. Springer - Verlag, 2005.

[79] E. W. Dijkstra, "Programming as a discipline of mathematical nature," *The American Mathematical Monthly*, vol. 81, no. 6, pp. 608–612, June-July 1974.

[80] ——, *A Discipline of Programming*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1976.

[81] S. Dodge, R. Weibel, and A.-K. Lautenschütz, "Towards a taxonomy of movement patterns," *Information Visualization*, vol. 7, no. 3-4, pp. 240–252, Autumn/Winter 2008.

[82] A. Drozdek, *Data Structures and Algorithms in C++*, 3rd ed. Course Technology, September 2004.

[83] F. Dylla, "Qualitative spatial reasoning for navigating agents," in *Behaviour Monitoring and Interpretation - BMI*, B. Gottfried and H. K. Aghajan, Eds. Amsterdam, Netherlands: IOS Press BV, 2009, pp. 98–128.

[84] M. J. Egenhofer, "A formal definition of binary topological relationships," *Lecture Notes in Computer Science*, vol. 367, pp. 457–472, June 1989.

[85] ——, "Reasoning about binary topological relations," in *Second International Symposium on Advances in Spatial Databases*. Springer-Verlag, 1991, pp. 143–160.

[86] ——, "Spherical topological relations," *Lecture Notes in Computer Science*, vol. 3534, pp. 25–49, 2005.

[87] M. J. Egenhofer and M. P. Dube, "Topological relations from metric refinements," in *Seventeenth ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Seattle, WA, USA: Association for Computing Machinery (ACM), November 2009, pp. 158–167.

[88] M. J. Egenhofer and A. U. Frank, "LOBSTER: Combining AI and database techniques for GIS," *Photogrammetric Engineering & Remote Sensing*, vol. 56, no. 6, pp. 919–926, June 1990.

[89] M. J. Egenhofer, A. U. Frank, and J. P. Jackson, “A topological data model for spatial databases,” *Lecture Notes in Computer Science*, vol. 409, pp. 271–286, July 1989.

[90] M. J. Egenhofer and R. D. Franzosa, “Point-set topological spatial relations,” *International Journal of Geographical Information Systems*, vol. 5, no. 2, pp. 161–174, 1991.

[91] ——, “A model for detailed binary topological relationships,” *Geomatica*, vol. 47, no. 3&4, pp. 261–273, 1993.

[92] ——, “On the equivalence of topological relations,” *International Journal of Geographical Information Systems*, vol. 9, no. 2, pp. 133–152, 1995.

[93] M. J. Egenhofer, J. Glasgow, O. Günther, J. R. Herring, and D. J. Peuquet, “Progress in computational methods for representing geographic concepts,” *International Journal of Geographical Information Systems*, vol. 13, no. 8, pp. 775–796, 1999.

[94] M. J. Egenhofer and J. R. Herring, “Categorizing binary topological relations between regions, lines, and points in geographic databases,” Department of Surveying Engineering, University of Maine, Orono, ME, USA, Tech. Rep., 1990.

[95] ——, “A mathematical framework for the definition of topological relationships,” in *Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, July 1990, pp. 803–813.

[96] M. J. Egenhofer and D. M. Mark, “Modeling conceptual neighborhoods of topological line-region relations,” *International Journal of Geographical Information Systems*, vol. 9, no. 5, pp. 555–565, 1995.

[97] M. J. Egenhofer and A. R. B. M. Shariff, “Metric details for natural-language spatial relations,” *ACM Transactions on Information Systems*, vol. 16, no. 4, pp. 295–321, October 1998.

[98] M. J. Egenhofer, J. Sharma, and D. M. Mark, “A critical comparison of the 4-intersection and 9-intersection models for spatial relations: Formal analysis,” in *International Symposium On Computer-Assisted Cartography (Auto-Carto-11)*, R. B. McMaster and M. P. Armstrong, Eds. Minneapolis, MN: American Society for Photogrammetry and Remote Sensing, October 1993, pp. 1–11.

[99] C. S. Ellis, “Concurrent search and insertions in 2-3 Trees,” *Acta Informatica*, vol. 14, no. 1, pp. 63–86, 1980.

[100] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, 1st ed. John Wiley & Sons, Inc., 2005.

[101] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Magazine.*, vol. 17, no. 3, pp. 37–54, Fall 1996.

[102] G. S. Fishman, *Principles of Discrete Event Simulation*, 1st ed. John Wiley & Sons, Inc., 1978.

[103] R. W. Floyd and J. D. Ullman, “The compilation of regular expressions into integrated circuits,” *Journal of the ACM*, vol. 29, no. 3, pp. 603–622, July 1982.

[104] S. Fogel, T. Morales, P. Potineni, and S. Moore, “Oracle 11g Release 1 (11.1): Administrator’s guide (B28310-04),” *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, March 2008.

[105] F. T. Fonseca, M. J. Egenhofer, P. Agouris, and G. Câmara, “Using ontologies for integrated geographic information systems,” *Transactions in GIS*, vol. 6, no. 3, pp.

231–257, 2002.

[106] F. T. Fonseca, M. J. Egenhofer, C. Davis, and G. Câmara, “Semantic granularity in ontology-driven geographic information systems,” *Annals of Mathematics and Artificial Intelligence*, vol. 36, no. 1-2, pp. 121–151, 2002.

[107] Foothills Research Institute, “Home webpage of the foothills research institute,” World Wide Web, http://foothillsresearchinstitute.ca/pages/home/, 1176 Switzer Drive. Hinton, AB T7V 1V3, Canada, January 2012.

[108] M.-J. Fortin and M. R. T. Dale, *Spatial Analysis: A Guide for Ecologists*, 1st ed. Cambridge University press, May 2005.

[109] J. L. Frair, S. E. Nielsen, E. H. Merrill, S. R. Lele, M. S. Boyce, R. H. M. Munro, G. B. Stenhouse, and H. L. Beyer, “Removing GPS collar bias in habitat selection studies,” *Journal of Applied Ecology, Published by the British Ecological Society*, vol. 41, no. 2, pp. 201–212, April 2004.

[110] S. E. Franklin, D. R. Peddle, J. A. Dechka, and G. B. Stenhouse, “Evidential reasoning with landsat TM, DEM and GIS data for landcover classification in support of Grizzly bear habitat mapping,” *International Journal of Remote Sensing*, vol. 23, no. 21, pp. 4633–4652, 2002.

[111] Y. Gao, Y. Zhang, Y. Tian, and J. N. Weng, “Topological relations between directed lines and simple geometries,” *Science in China, Series E: Technological Sciences*, vol. 51, pp. 91–101, 2008.

[112] D. L. Garshelis, M. L. Gibeau, and S. Herrero, “Grizzly bear demographics in and around Banff National Park and Kananaskis country, Alberta,” *Journal of Wildlife Management*, vol. 69, no. 1, pp. 277–297, January 2005.

[113] A. Gibbons, *Algorithmic Graph Theory*, 1st ed. Cambridge, UK: Cambridge University Press, July 1985.

[114] M. L. Gibeau, “A conservation biology approach to management of Grizzly bears in Banff National Park, Alberta,” Ph.D. dissertation, University of Calgary, Calgary, Alberta, Canada, April 2000.

[115] M. L. Gibeau, A. P. Clevenger, S. Herrero, and J. Wierzchowski, “Grizzly bear response to human development and activities in the Bow river watershed, Alberta, Canada,” *Biological Conservation*, vol. 103, pp. 227–236, 2002.

[116] M. L. Gibeau, S. Herrero, B. N. McLellan, and J. G. Woods, “Managing for Grizzly bear security areas in Banff National Park and the central Canadian Rocky Mountains,” *Ursus*, vol. 12, pp. 121–130, 2001.

[117] O. Goldreich, *P, NP, and NP-completeness: The basics of computational complexity*, 1st ed. Cambridge University Press, August 2010.

[118] M. G. Gouda and L. E. Rosier, “Priority networks of communicating finite state machines,” *SIAM Journal on Computing*, vol. 14, no. 3, pp. 569–584, 1985.

[119] D. Grune and C. J. H. Jacobs, *Parsing Techniques: A Practical Guide*, 1st ed., ser. Monographs in Computer Science. Berlin, Germany: Springer-Verlag, 2008.

[120] J. Gudmundsson, M. J. van Kreveld, and B. Speckmann, “Efficient detection of motion patterns in spatio-temporal data sets,” in *Proceedings of the Twelfth Annual ACM International Workshop on Geographic Information Systems (ACM)*, Conference on Information and Knowledge Management (CIKM). New York, NY, USA: Association for Computing Machinery (ACM), 2004, pp. 250–257.

[121] L. J. Guibas, E. M. McCreight, M. F. Plass, and J. R. Roberts, "A new representation for linear lists," in *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery (ACM), 1977, pp. 49–60.

[122] L. J. Guibas, J. S. B. Mitchell, and T. Roos, "Voronoi diagrams of moving points in the plane," *Lecture Notes in Computer Science*, vol. 570, pp. 113–125, 1992.

[123] L. J. Guibas and A. M. Odlyzko, "A new proof of the linearity of the Boyer-Moore string searching algorithm," in *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science (ACM-SIAM)*. Providence, RI, USA: SIAM: Society for Industrial and Applied Mathematics, November 1997, pp. 189–195.

[124] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer science and computational biology*, 1st ed. New York, NY, USA: Cambridge University Press, 1997.

[125] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM Special Interest Group on Management of Data (SIGMOD), International Conference on Management of Data*, vol. 14(2). New York, NY, USA: Association for Computing Machinery (ACM), June 1984, pp. 47–57.

[126] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 26, no. 2, pp. 147–160, April 1950.

[127] ——, *Coding and Information Theory*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1980.

[128] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann (Elsevier), 2006.

[129] S. J. Hannon and C. McCallum, "Using the focal species approach for conserving biodiversity in landscapes managed for forestry," *Sustainable Forest Management Network*, 2004.

[130] D. C. Heard, L. M. Ciarniello, and D. R. Seip, "Grizzly bear behavior and global positioning system collar fix rates," *Journal of Wildlife Management*, vol. 72, no. 3, pp. 596–602, April 2008.

[131] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, pp. 10 915–10 919, November 1992.

[132] D. Hinrichsen and A. J. Pritchard, *Mathematical Systems Theory I: Modelling, state space analysis, stability and robustness*, 1st ed., ser. Texts in Applied Mathematics. Berlin, Germany: Springer-Verlag, 2010, vol. 48.

[133] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces," *ACM Transactions on Database Systems (TODS)*, vol. 28, no. 4, pp. 517–580, December 2003.

[134] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd ed. Reading, MA, USA: Addison Wesley, 2001.

[135] K. Hornsby and M. J. Egenhofer, "Identity-based change: A foundation for spatio-temporal knowledge representation," *International Journal of Geographical Information Systems*, vol. 14, no. 3, pp. 207–224, 2000.

[136] ——, "Modeling moving objects over multiple granularities," *Annals of Mathematics and Artificial Intelligence*, vol. 36, no. 1-2, pp. 177–194, 2002.

[137] R. N. Horspool, "Practical fast searching in strings," *Software: Practice and Experience*, vol. 10, no. 6, pp. 501–506, June 1980.

[138] H. Y. Huang and J. P. Chamblis, "Quadratically convergent algorithms and one-dimensional search schemes," *Journal of Optimization Theory and Applications*, vol. 11, no. 2, pp. 175–188, 1973.

[139] A. J. S. Hunter, "Sensor-based animal tracking," Ph.D. dissertation, University of Calgary, Calgary, Alberta, Canada, September 2007.

[140] F. Javed, "Techniques for context-free grammar induction and applications," Ph.D. dissertation, The University of Alabama at Birmingham, Birmingham, AL, USA, 2008.

[141] ——, *Techniques for Context-Free Grammar Induction and Applications: Application of novel inference algorithms to software maintenance problems*, 1st ed. VDM Verlag, January 2009.

[142] S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 87, no. 6, pp. 2264–2268, March 1990.

[143] S. Karlin, A. Dembo, and T. Kawabata, "Statistical composition of high scoring segments from molecular sequences," *The Annals of Statistics*, vol. 18, no. 2, pp. 571–581, June 1990.

[144] P. L. Karlton, S. H. Fuller, R. E. Scroggs, and E. B. Kaehler, "Performance of height-balanced trees," *Communications of the ACM*, vol. 19, no. 1, pp. 23–28, January 1976.

[145] D. Kerzel and W. Prinz, "Performance," in Encyclopedia of Cognitive Science, John Wiley & Sons, Inc. http://onlinelibrary.wiley.com/doi/10.1002/0470018860.s00592/pdf, January 2006.

[146] D. E. Knuth, *The art of computer programming*, 1st ed. Addison Wesley, 1974.

[147] D. E. Knuth, J. H. Morris (Jr), and V. R. Pratt, "Fast pattern matching in strings," *Stanford University: Computer Science Department Reports*, vol. STAN-CS-74-440, 1974.

[148] ——, "Fast pattern matching in strings," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 323–350, June 1977.

[149] L. Kulik, M. Duckham, and M. J. Egenhofer, "Ontology-driven map generalization," *Journal of Visual Languages and Computing*, vol. 16, no. 3, pp. 245–267, 2005.

[150] A. Küpper, *Location-Based Services: Fundamentals and operation*, 1st ed. John Wiley & Sons, Inc., October 2005.

[151] Y. Kurata, "The 9^+-intersection: A universal framework for modeling topological relations," *Lecture Notes in Computer Science*, vol. 5266, pp. 181–198, 2008.

[152] ——, "A strategy for drawing a conceptual neighborhood diagram schematically," *Lecture Notes in Computer Science*, vol. 5223, pp. 388–390, 2008.

[153] Y. Kurata and M. J. Egenhofer, "The head-body-tail intersection for topological relations between directed line segments," *Lecture Notes in Computer Science*, vol. 4197, pp. 269–286, 2006.

[154] ——, "The 9^+-intersection for topological relations between a directed line segment

and a region," Technologie-Zentrum Informatik, University of Bremen, Germany, Tech. Rep., 2007.

[155] ——, "Interpretation of behaviours from a viewpoint of topology," in *Behaviour Monitoring and Interpretation - BMI*, B. Gottfried and H. K. Aghajan, Eds. Amsterdam, Netherlands: IOS Press BV, 2009, pp. 75–97.

[156] Y.-S. Kwong and D. Wood, "A new method for concurrency in B-Trees," *IEEE Transactions on Software Engineering*, vol. 8, no. 3, pp. 211–222, May 1982.

[157] R. Larson and D. C. Falvo, *Elementary Linear Algebra*, 6th ed. Pacific Grove, CA, USA: Brooks Cole, July 2008.

[158] P. Laube, "Progress in movement pattern analysis," in *Behaviour Monitoring and Interpretation - BMI*, B. Gottfried and H. K. Aghajan, Eds. Amsterdam, Netherlands: IOS Press BV, 2009, pp. 43–71.

[159] P. Laube and S. Imfeld, "Analyzing relative motion within groups of trackable moving point objects," *Lecture Notes in Computer Science*, vol. 2478, pp. 132–144, 2002.

[160] P. Laube, R. S. Purves, S. Imfeld, and R. Weibel, *Analysing point motion with geographic knowledge discovery techniques*, 1st ed., ser. Innovations in GIS. Dynamic and Mobile GIS: Investigating Change in Space and Time. Boca Raton, FL, USA: CRC press. Taylor & Francis, 2007, pp. 263–286.

[161] P. Laube, M. J. van Kreveld, and S. Imfeld, "Finding REMO - detecting relative motion patterns in geospatial lifelines," in *Developments in Spatial Data Handling: Eleventh International Symposium on Spatial Data Handling.* Heidelberg: Springer-Verlag, October 2004, pp. 201–214.

[162] P. N. Laver and M. J. Kelly, “A critical review of home range studies,” *Journal of Wildlife Management*, vol. 72, no. 1, pp. 290–298, 2008.

[163] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, February 1966.

[164] Y. Li, J. Han, and J. Yang, “Clustering moving objects,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD). Seattle, WA, USA: Association for Computing Machinery (ACM), August 2004, pp. 617–622.

[165] D. B. Lindenmayer, C. R. Margules, and D. B. Botkin, “Indicators of biodiversity for ecologically sustainable forest management,” *Conservation Biology*, vol. 14, no. 4, pp. 941–950, 2000.

[166] J. Linke, S. E. Franklin, F. Huettmann, and G. B. Stenhouse, “Seismic cutlines, changing landscape metrics and grizzly bear landscape use in Alberta,” *Landscape Ecology*, vol. 20, no. 7, pp. 811–826, 2005.

[167] D. J. Lipman and W. R. Pearson, “Rapid and sensitive protein similarity searches,” *Science*, vol. 227, no. 4693, pp. 1435–1441, 1985.

[168] E. H.-C. Lu and V. S. Tseng, “Mining cluster-based mobile sequential patterns in location-based service environments,” in *Proceedings of the Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, Taipei, Taiwan, May 2009, pp. 273–278.

[169] M. Main and W. Savitch, *Data Structures and Other Objects Using C++*, 4th ed. Addison Wesley, March 2010.

[170] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and applications*, 1st ed. London: Springer - Verlag, 2006.

[171] K. Maruyama and S. E. Smith, "Analysis of design alternatives for virtual memory indexes," *Communications of the ACM*, vol. 20, no. 4, pp. 245–254, April 1977.

[172] E. M. McCreight, "Pagination of B*-Trees with variable-length records," *Communications of the ACM*, vol. 20, no. 9, pp. 670–674, September 1977.

[173] B. N. McLellan and F. W. Hovey, "Habitats selected by Grizzly bears in a multiple use landscape," *The Journal of Wildlife Management*, vol. 65, no. 1, pp. 92–99, January 2001.

[174] A. Meduna, *Automata and Languages: Theory and applications*, 1st ed. London, UK: Springer - Verlag, August 2000.

[175] A. Meduna and M. Švec, *Grammars with Context Conditions and Their Applications*, 1st ed. West Sussex, UK: Wiley - Interscience, June 2005.

[176] B. Miller, R. Reading, J. Strittholt, C. Carroll, R. Noss, M. Soule, O. Sanchez, J. Terborgh, D. Brightsmith, T. Cheeseman, and D. Foreman, "Focal species in the design of nature reserve networks," *Wild Earth Special Issues*, pp. 81–92, 1998.

[177] H. J. Miller, "A measurement theory for time geography," *Geographical Analysis*, vol. 37, no. 1, pp. 17–45, 2005.

[178] H. J. Miller and S. A. Bridwell, "A field-based theory for time geography," *Annals of the Association of American Geographers*, vol. 99, no. 1, pp. 49–75, 2009.

[179] H. J. Miller and J. Han, *Geographic Data Mining and Knowledge Discovery*, 2nd ed. Boca Raton, FL, USA: CRC press. Taylor & Francis, 2009.

[180] A. B. Molitor, “The effects of persuasive communication messages on backcountry visitor behavior in occupied Grizzly bear habitat,” Ph.D. dissertation, University of Montana, Missoula, MT, USA, April 1995.

[181] T. K. Moon, *Error Correction Coding: Mathematical methods and algorithms*, 1st ed. John Wiley & Sons, Inc., May 2005.

[182] D. R. Morrison, “PATRICIA - Practical algorithm to retrieve information coded in alphanumeric,” *Journal of the Association for Computing Machinery*, vol. 15, no. 4, pp. 514–534, October 1968.

[183] R. H. M. Munro, S. E. Nielsen, M. H. Price, G. B. Stenhouse, and M. S. Boyce, “Seasonal and diel patterns of Grizzly bear diet and activity in West-central Alberta,” *Journal of Mammalogy*, vol. 87, no. 6, pp. 1112–1121, 2006.

[184] Natural Regions Committee, “Natural regions and subregions of Alberta,” World Wide Web, http://www.cd.gov.ab.ca/preserving/parks/anhic/Natural_region_report.asp, Compiled by Downing, D.J. and Pettapiece, W.W. Government of Alberta, Edmonton, Tech. Rep. T/852, 2006.

[185] G. Navarro, “A guided tour to approximate string matching,” *Computing Surveys*, vol. 33, no. 1, pp. 31–88, March 2001.

[186] K. A. Nedas, M. J. Egenhofer, and D. Wilmsen, “Metric details of topological line-line relations,” *International Journal of Geographical Information Systems*, vol. 21, no. 1, pp. 21–48, 2007.

[187] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, March 1970.

[188] S. E. Nielsen, “Habitat ecology, conservation, and projected population viability of Grizzly bears (Ursus arctos L.) in West-central Alberta, Canada,” Ph.D. dissertation, University of Alberta, Edmonton, Alberta, Canada, 2005.

[189] S. E. Nielsen, M. S. Boyce, and G. B. Stenhouse, “Grizzly bears and forestry I. Selection of clearcuts by Grizzly bears in West-central Alberta, Canada,” *Forest Ecology and Management*, vol. 199, pp. 51–65, 2004.

[190] S. E. Nielsen, G. McDermid, G. B. Stenhouse, and M. S. Boyce, “Dynamic wildlife habitat models: Seasonal foods and mortality risk predict occupancy-abundance and habitat selection in Grizzly bears,” *Biological Conservation*, vol. 143, pp. 1623–1634, 2010.

[191] J. Nievergelt, “Binary search trees and file organization,” *Computing Surveys*, vol. 6, no. 3, pp. 195–207, September 1974.

[192] R. Nola and H. Sankey, *Theories of Scientific Method : An introduction*, 1st ed. McGill-Queen’s University Press, November 2007.

[193] R. F. Noss, “Indicators for monitoring biodiversity: A hierarchical approach,” *Conservation Biology*, vol. 4, no. 4, pp. 355–364, December 1990.

[194] H. Nyquist, “Certain topics in telegraph transmission theory,” *Transactions of the A. I. E. E.*, vol. 47, pp. 617–644, April 1928, reprinted in Proceedings of the IEEE, Vol. 90, No. 2, Feb 2002.

[195] S. Openshaw and R. J. Abrahart, *GeoComputation*, 1st ed. London, UK: CRC Press, Taylor & Francis, 2000.

[196] Oracle Corporation, “SPARC T3-1B Server Module (ds-173106),” *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, 2011.

[197] ——, "Sun Blade T6320 Server Module (033616)," *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, 2011.

[198] ——, "Sun Blade T6340 Server Module (034667)," *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, 2011.

[199] ——, "Sun Blade X6275 M2 Server Module (ds-182643)," *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, 2011.

[200] T. Ottmann and D. Wood, "1-2 brother trees or AVL trees revisited," *The Computer Journal*, vol. 23, no. 3, pp. 248–255, 1980.

[201] M. H. Overmars and J. Van Leeuwen, "Maintenance of configurations in the plane," *Journal of Computer and System Sciences*, vol. 23, no. 2, pp. 166–204, October 1981.

[202] D. Papadias, T. Sellis, Y. Theodoridis, and M. J. Egenhofer, "Topological relations in the world of minimum bounding rectangles: A study with R-Trees," in *Proceedings of the 1995 ACM Special Interest Group on Management of Data (SIGMOD), International Conference on Management of Data*, vol. 24(2). San Jose, CA, USA: Association for Computing Machinery (ACM), May 1995, pp. 92–103.

[203] A. D. Pape and S. E. Franklin, "MODIS-based change detection for Grizzly bear habitat mapping in Alberta," *Journal of Photogrammetric Engineering & Remote Sensing*, vol. 74, no. 8, pp. 973–985, August 2008.

[204] L. Parrott and R. Kok, "Incorporating complexity in ecosystem modeling," World Wide Web, Complexity International 7: http://www.complexity.org.au/, 2000.

[205] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences of the United States of*

America, vol. 85, no. 8, pp. 2444–2448, April 1988.

[206] F. Qi and A.-X. Zhu, “Knowledge discovery from soil maps using inductive learning,” *International Journal of Geographical Information Science*, vol. 17, no. 8, pp. 771–795, 2003.

[207] T. Raita, “Tuning the Boyer-Moore-Horspool string searching algorithm,” *Software: Practice and Experience*, vol. 22, no. 10, pp. 879–884, October 1992.

[208] R. M. P. Reis, M. J. Egenhofer, and J. L. G. Matos, “Conceptual neighborhoods of topological relations between lines,” *Lecture Notes in Geoinformation and Cartography*, pp. 557–574, 2008.

[209] A. Rheinländer, M. Knobloch, N. Hochmuth, and U. Leser, “Prefix tree indexing for similarity search and similarity joins on genomic data,” *Lecture Notes in Computer Science*, vol. 6187, pp. 519–536, 2010.

[210] A. Rheinländer and U. Leser, “Fast similarity searches and similarity joins in Oracle DB,” in *The German Oracle Users Group Conference (DOAG).* Nürnberg, Germany: Oracle Corporation, November 2010.

[211] S. Rizzi and M. Golfarelli, *Data Warehouse Design: Modern principles and methodologies*, 1st ed. Emeryville, CA, USA: McGraw-Hill Osborne Media, May 2009.

[212] P. Rob and C. Coronel, *Database Systems: Design, implementation, and management*, 8th ed. Cambridge, MA, USA: Course Technology, December 2007.

[213] W. S. Robinson, “Ecological correlations and the behavior of individuals,” *American Sociological Review*, vol. 15, pp. 351–357, June 1950.

[214] A. M. Rodríguez and M. J. Egenhofer, “Determining semantic similarity among

entity classes from different ontologies," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 2, pp. 442–456, 2003.

[215] ——, "Comparing geospatial entity classes: An asymmetric and context-dependent similarity measure," *International Journal of Geographical Information Science*, vol. 18, no. 3, pp. 229–256, April-May 2004.

[216] C. L. Roever, M. S. Boyce, and G. B. Stenhouse, "Grizzly bears and forestry I: Road vegetation and placement as an attractant to Grizzly bears," *Forest Ecology and Management*, vol. 256, no. 6, pp. 1253–1261, 2008.

[217] ——, "Grizzly bear movements relative to roads: Application of step selection functions," *Ecography*, vol. 33, pp. 1113–1122, December 2010.

[218] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19–26, 1989.

[219] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," in *Proceedings of the 2000 ACM Special Interest Group on Management of Data (SIGMOD), International Conference on Management of Data*, vol. 29(2). New York, NY, USA: Association for Computing Machinery (ACM), June 2000, pp. 331–342.

[220] H. Samet, *Foundations of Multidimensional and Metric Data Structures*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann (Elsevier), September 2006.

[221] A. A. Schäffer, L. Aravind, T. L. Madden, S. Shavirin, J. L. Spouge, Y. I. Wolf, E. V. Koonin, and S. F. Altschul, "Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements," *Nucleic Acids Research*, vol. 29, no. 14, pp. 2994–3005, July 2001.

[222] P. H. Sellers, “On the theory and computation of evolutionary distances,” *SIAM Journal on Applied Mathematics*, vol. 26, no. 4, pp. 787–793, 1974.

[223] C. E. Shannon, “Communication in the presence of noise,” *Proceedings of the Institute of Radio Engineers*, vol. 37, no. 1, pp. 10–21, January 1949, reprinted in Proceedings of the IEEE, Vol. 86, No. 2, Feb 1998.

[224] H. Shi and Y. Kurata, “Modeling ontological concepts of motions with two projection-based spatial models,” in *Second Workshop on Behavioral Monitoring and Interpretation*, B. Gottfried and H. K. Aghajan, Eds., vol. 396. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 42–56.

[225] ——, “Modeling ontological concepts of locations with a heterogeneous cardinal direction model,” *International Journal of Software and Informatics,*, vol. 4, no. 3, pp. 239–256, September 2010.

[226] A. Singh, *Elements of Computation Theory*, 1st ed., ser. Texts in Computer Science. Berlin, Germany: Springer-Verlag, 2009, vol. 47.

[227] K. I. Smith, R. M. Everson, J. E. Fieldsend, C. Murphy, and R. Misra, “Dominance-based multiobjective simulated annealing,” *IEEE Transactions on Studies in Computational Intelligence*, vol. 12, no. 3, pp. 323–342, June 2008.

[228] T. F. Smith and M. S. Waterman, “Identification of common molecular subsequences,” *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, March 1981.

[229] W. Stallings, *Operating Systems: Internals and Design Principles*, 6th ed. Prentice Hall, April 2008.

[230] ——, *Computer Organization and Architecture: Designing for Performance*, 8th ed. Prentice Hall, April 2009.

[231] F. Stomp, “Correctness of substring-preprocessing in Boyer-Moore’s pattern matching algorithm,” *Software: Practice and Experience*, vol. 290, no. 1, pp. 59–78, January 2003.

[232] C. Strobl, “Dimensionally extended nine-intersection model (DE-9IM),” in *Encyclopedia of GIS*, S. Shekar and H. Xiong, Eds. Springer Science, Business Media, LLC., 2008, pp. 240–245.

[233] W. L. Strong, *Ecoregions and Ecodistricts of Alberta*. Edmonton, Alberta: Alberta Forestry, Lands and Wildlife. Land Information Services Division, 1992, vol. 1.

[234] S. Sumathi and S. Esakkirajan, *Fundamentals of Relational Database Management Systems*, 1st ed., ser. Studies in Computational Intelligence. Berlin, Germany: Springer-Verlag, 2007, vol. 47.

[235] D. M. Sunday, “A very fast substring search algorithm,” *Communications of the ACM*, vol. 33, no. 8, pp. 132–142, August 1990.

[236] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed. Prentice Hall, December 2007.

[237] M. Thériault, C. Claramunt, and P. Y. Villeneuve, “A spatio-temporal taxonomy for the representation of spatial set behaviours,” *Lecture Notes in Computer Science*, vol. 1678, pp. 1–18, 1999.

[238] T. Timmins, “Modelling spatial dependence in multivariate regression models of grizzly bear health in Alberta, Canada,” Master’s thesis, University of Calgary, Calgary, Alberta, Canada, 2010.

[239] W. R. Tobler, “A computer movie simulating urban growth in the Detroit region,” *Economic Geography*, vol. 46, no. 2, pp. 234–240, June 1970.

[240] L. T. Toth, “The feasibility of classifying Grizzly bear habitat in the Cascade Valley, Banff National Park, Alberta, by digital analysis of Landsat-2 multispectral scanner data,” Master’s thesis, University of Calgary, Calgary, Alberta, Canada, 1985.

[241] N. Tryfona and M. J. Egenhofer, “Multi-resolution spatial databases: Consistency among networks,” in *Integrity in Databases - Sixth International Workshop on Foundations of Models and Languages for Data and Objects*, S. Conrad, H.-J. Klein, and K.-D. Schewe, Eds., Schloss Dagstuhl, Germany, September 1996, pp. 119–132.

[242] V. S. Tseng and C.-P. Kao, “Mining and validating gene expression patterns: An integrated approach and applications,” *Informatica*, vol. 27, no. 1, pp. 21–27, April 2003.

[243] ——, “Efficiently mining gene expression data via a novel parameterless clustering method,” *ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 4, pp. 355–365, October-December 2005.

[244] R. Urbano, N. Arora, S. Balaraman, Y. Chan, A. Downing, C. Elsbernd, Y. Feng, J. Galagali, L. Kaplan, J. Klein, J. Liu, E. Lu, P. McElroy, M. Pratt, A. Rajaram, N. Shodhan, W. Smith, J. Stamos, J. Stern, M. Subramaniam, L. Wong, and D. Zhang, “Oracle 11g Release 1 (11.1): Advanced replication (b28326-03),” *Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, USA*, August 2008.

[245] D. J. Wilbur, W. John; Lipman, “Rapid similarity searches of nucleic acid and protein data banks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 80, no. 3, p. 726730, February 1983.

[246] Z. Wood and A. Galton, “Classifying collective motion,” in *Behaviour Monitoring and Interpretation - BMI*, B. Gottfried and H. K. Aghajan, Eds. Amsterdam, Netherlands: IOS Press BV, 2009, pp. 129–155.

[247] R. Wrembel and C. Koncilia, *Data warehouses and OLAP: Concepts, architectures, and solutions*, 1st ed. Hershey, PA, USA: IGI Global, 2007.

[248] X. Ye, G. Wang, and S. F. Altschul, "An assessment of substitution scores for protein profile-profile comparison," *Bioinformatics*, October 2011.

[249] H. D. Young, R. A. Freedman, and A. L. Ford, *University Physics with Modern Physics*, 13rd ed. Addison Wesley, January 2011.

[250] Y.-K. Yu and S. F. Altschul, "The complexity of the Dirichlet model for multiple alignment data," *Journal of Computational Biology*, vol. 18, no. 8, pp. 925–939, 2011.

[251] C.-H. Yun and M.-S. Chen, "Mining mobile sequential patterns in a mobile commerce environment," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 2, pp. 278–295, March 2007.

[252] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, November 1999.

Appendix A

Initial analysis methods.

As part of the research activities of the current work, several computerized solution methods were considered (e.g. Voronoi polygon analysis), and the two included in this appendix, the LBS and the simulated annealing methods, were explored to the point of implementation, to determine their effectiveness as an analysis tool for the grizzly bear tracking data in this project (i.e. fully integrated and with reasonable performance and scalability).

The fully integrated implementation details shown in this appendix highlight some of the flexibility and robustness of a contemporary RDBMS platform. However, the individual technical and/or conceptual limitations of these two methods, as detailed in sections A.1.2 and A.2.2, rendered them as unsuitable for the purposes of this work.

A.1 Location-Based Services (LBS).

In order to provide a definition of LBS, the text by Küpper [150] starts by saying:

> "Although *Location-Based Services* (LBSs) have been an issue in the field of mobile communications for many years, there exists neither a common definition nor a common terminology for them. For example, the term *location-based service*, *location-aware service*, *location-related service*, and *location service* are often interchangeably used. One reason for this dilemma might lie in the fact that the character and appearance of such services have been determined by different communities, especially the telecommunications sector and the ubiquitous computing area."

Clarifying, the author continues:

> "LBSs are always context-aware services... LBSs can be classified into *reactive* and *proactive LBSs*. A reactive LBS is always explicitly activated by the user... Proactive LBSs, on the other hand, are automatically initialized as soon as a predefined location event occurs, for example, if the user enters, approaches, or leaves a certain point of interest or if he approaches, meets, or leaves another target."

Providing in essence an operational definition of the term and its specialized variations, with the use of the concepts of events in time (i.e. instantaneous events), and proximity to predefined locations (i.e. metric proximity), as expressions of the context of the events themselves.

In the case of the current work, the concept of events in a specific instant in time was not analyzed, and the rest of this document considers computational techniques from the LBS specialty exclusively for their technical ability to analyze large quantities of trajectories in order to identify similar subtrajectories.

Many studies of human movement patterns in urban environments monitor the activities of individual subjects through temporally and geographically tagged service use logs [164, 168, 177, 178, 179, 243, 251]. In some of those studies, the geographical extension is restricted to a relatively small regular grid of predefined locations of interest (e.g. the grid of wireless telephony access points at a shopping mall). This restriction is the equivalent of a domain transformation from the geographical extension to an abstract classification. The spatial resolution and total extension of the grid directly determines the maximum level of detail that can be captured by the monitoring process, and indirectly reduces the overall length of all trajectories. In most cases, the services monitored in the grid correspond to information sources, both public (e.g. the public bus schedule serving the venue) and commercial (e.g. the showing's schedule of a nearby movie theater). This approach of activity monitoring and its studies are part of the specialty known as LBS [150]. Figure A.1 shows a simple example of an LBS grid with two different trajectories on it (*T1* and *T2*), the horizontal $x - y$ plane represents geographic position (with the LBS grid defined by the hexagons), while the vertical axis represents the time dimension.

Drawing an analogy to LBS but with the purpose of detecting movement patterns in grizzly bear tracking datasets, the grid can be replaced by a digital map with a regular tessellation (e.g. a digital elevation map). As section 2.1 identified, relevant grizzly bear data can be used to substitute the human services, and include land cover maps, digital elevation maps, distance to roads maps, seasonal food availability maps, etc. In other words, maps of localized features that attract or repel grizzly bears [39, 116, 216]. One important limitation of this approach is the spatial resolution of the abstract classification maps, and their associated range of classes, which is very narrow compared to all possible values for geographical distance, and represents a degradation of spacial resolution.

Continuing towards a more abstract definition of patterns, Tseng et al. (e.g. [242, 243, 168]) have dedicated several works to the study of sub-trajectories of similar characteristics. Their early studies come from the biological sciences, where ordered molecule sequences in chemistry and genetics are of paramount importance. One of their early papers [242], describes the validity of data mining methods applied to gene expression sequences. Their approach considers just the abstract class that each molecule gets assigned (i.e., X or O) and the absolute order of the sub-sequences.

Some of their latest LBS work [168] however, employs a complicated multi-criteria evaluation of both, geographic and abstract information. Their method includes the initial segmentation of trajectories into individual points in space-time, and the generation of common portions of multiple points by a maximum support method based

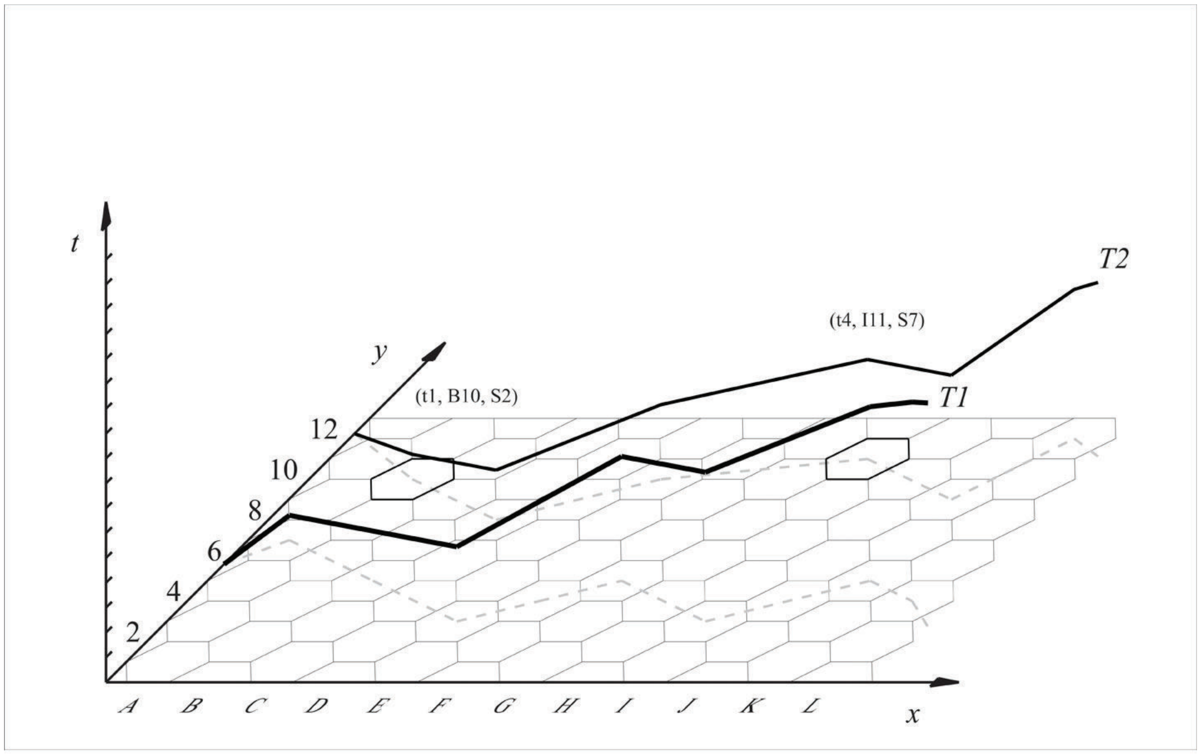

Figure A.1: Location-based services, including trajectory concepts.

simultaneously on geographical location and service (similar to a voting method). Their sub-trajectory similarity function incorporates a custom normalization for each pair of sub-trajectories, based on their total number of locations. The rest of the criteria elements include reward factors for location and service, and one penalty factor for time difference between points, as measured against a global time origin for their analysis. Their algorithm also compensates for the tendency of the compared sub-trajectories to be mostly different rather than similar by using Rand and Jaccard's coefficient approaches ([243], pg. 359, [168], pg. 276-277). Additional to their first design, indexing structures were added to their methods to speed up the process.

A.1.1 Implementation.

Because of the flexibility of its multicriteria integration and its relative simplicity (section A.1), a modified implementation of the algorithm proposed by Lu and Tseng [168] (page 274) was employed in this work. Algorithm A.1 shows the modified version of the method. The changes to the original were conducted in the form of factor substitutions or additions, as explained next.

Factor substitutions:

- Gender reward replaced service reward.
- Land cover agreement took the place of service agreement.
- Time of the day shift penalty replaced time shift penalty.
- Land cover frequency reward replaced service frequency reward.

In the original publication, service reward was applied with a set of only 5 services, the best equivalent factor in the current dataset was gender because it is a general property of each trajectory, further reducing the set to only two classes, female and male.

For both algorithms, the agreement decision determines a potentially large number of additional calculations. The set of 11 land cover classes was determined to better reflect the character of the original decision made with services. The same reason justified the substitution of the service frequency reward for the land cover frequency reward, calculated the same way in both versions.

In both algorithms the temporal component of the data is used in the same way, their only difference is their respective temporal origin points. For the original version, an absolute origin was defined for the entire database, while the current dataset defined the start of every 24 hour period as exactly equivalent.

The following penalty factors were added to the similarity calculation in an effort to increase the comprehensiveness of the method and to better reflect the geographical

Algorithm A.1 Modified LBS-Alignment similarity measure. Based on [168].

```
Input: Two mobile transaction sequences $s$ and $s'$
Output: The similarity between $s$ and $s'$
 1: $p \leftarrow 0.5/(s.Length + s'.Length)$                                   ▷ Penalty.
 2: $M_{0,0} \leftarrow 0.5$
 3: for $i \leftarrow 1$ to $s.Length$ do
 4:     $M_{i,0} \leftarrow M_{i-1,0} - p$
 5: end for
 6: for $j \leftarrow 1$ to $s'.Length$ do
 7:     $M_{0,j} \leftarrow M_{0,j-1} - p$
 8: end for
 9: $Max\_Distance \leftarrow Calc\_Max\_Distance(s, s')$
10: $Max\_Elevation\_Diff \leftarrow Calc\_Max\_Elevation\_Diff(s, s')$
11: if $s.Gender = s'.Gender$ then
12:     $Gender\_Reward \leftarrow p$
13: else
14:     $Gender\_Reward \leftarrow p * (-1)$
15: end if
16: for $i \leftarrow 1$ to $s.Length$ do
17:     for $j \leftarrow 1$ to $s'.Length$ do
18:         if $s_i.LandCover = s'_j.LandCover$ then
19:             $Location\_Penalty \leftarrow p * (Calc\_Distance(s_i, s'_j)/Max\_Distance)$
20:             $Time\_Penalty \leftarrow p * (\mid s'_j.Time - s_i.Time \mid /Time\_Length)$
21:             $Elevation\_Penalty \leftarrow p * (\mid s'_j.Elevation - s_i.Elevation \mid /Max\_Elevation\_Diff)$
22:             $Distance\_To\_Roads\_Penalty \leftarrow Calc\_Distance\_Penalty(p, s_i, s'_j)$
23:             $Total\_Penalty \leftarrow Location\_Penalty - Time\_Penalty - Elevation\_Penalty -$
24:                                  $Distance\_To\_Roads\_Penalty$
25:
26:             $Individual\_Counts \leftarrow Ind\_Counts[s_i.LandCover] * Ind\_Counts[s'_j.LandCover]$
27:             $LandCover\_Freq\_Reward \leftarrow p * (Pair\_Counts[s_i.LandCover, s'_j.LandCover]/Individual\_Counts)$
28:             $Total\_Reward \leftarrow LandCover\_Freq\_Reward + Gender\_Reward$
29:
30:             $M_{i,j} \leftarrow Max((M_{i-1,j} - p), (M_{i,j-1} - p), (M_{i-1,j-1} - Total\_Penalty + Total\_Reward))$
31:         else
32:             $M_{i,j} \leftarrow Max((M_{i-1,j} - p), (M_{i,j-1} - p))$
33:         end if
34:     end for
35: end for
36: return $M_{s.Length, s'.Length}$
```

character of the current dataset. The relevance of each of them has been examined in more detail in section 2.1.

- Location.
- Elevation.
- Distance to roads.

A.1.2 Results.

Initial execution with the calibration dataset (section 3.5) finished in 92 seconds, then it was decided to apply the method to the entire dataset. See subsection A.1.1 for implementation details.

Total execution time for one repetition of the method was 173.2 hours (7 days, 5.2 hours).

This can be attributed to the complexity of the multicriteria formula chosen to analyze this case, and the dynamic nature of the programming solution. See algorithm A.1 for the source of the following calculation.

For a pair of average length trajectories, the execution time for this method is directly proportional to the cost of calculating the maximum distance between trajectories **plus** the cost of calculating the maximum elevation difference between the pairs of trajectories **plus** (the square of the average trajectory length **times** the number of floating point arithmetic operations needed to evaluate the multicriteria formula). With an average trajectory length of 1778 points this would be, $c_1 = (1778)^2 + (1778)^2 + [(1778)^2 * 18)] = (1778)^2 * 20 = 63,225,680$ floating point operations. For the set of 208 average length trajectories, the entire number of floating point operations to process is given by equation A.1 [78].

$$c = c_1 \sum_{i=1}^{207} i = (63,225,680) * (21,528) = 1,361,122,439,040 \tag{A.1}$$

The second term in equation A.1 represents the comparison of every trajectory against every other trajectory in the set. See algorithm A.1 for additional details.

The dynamic nature of the algorithm and the complexity of its multicriteria formula, were judged significant enough limitations to determine this method as unsuitable for this dataset.

A.2 Simulated annealing.

The original method of simulated annealing comes from the numeric optimization specialty in mathematics. Simulated annealing optimization was originally designed to search for approximate solutions to Non-deterministic Polynomial Complete (NP-Complete) problems. In essence, the method comprises a searching algorithm that follows the slope of a function until an optimal point (minimum or maximum) is reached in a set amount of iterations (and time in some cases). To avoid local minima (or maxima) the algorithm uses a random step to switch the function's searched area, and accept or reject the new solution based on a decreasing threshold that represents an abstract temperature range. The objective of simulated annealing is not to arrive at an exact solution, but rather to a reasonable approximation according to a given maximum number of iterations and an acceptable error margin.

In the work of Rutenbar [218] the method was presented in full detail, analyzed for its non-deterministic and probabilistic implications, and exemplified with a pair of implementations for integrated circuit components' placement (i.e. internal microchip "floorplan"). The paper reminded the reader that the applicability of simulated annealing is not only limited by its mathematical aspects, but also by the developer's choices in the conceptual representation of its operational parts. In practice, the number of iterations and the appropriate abstract temperature range have to be determined in a case-by-case basis, converting them in additional potential limitations. Comparing simulated annealing with heuristic-based methods, the work stated that, a badly conceptualized or implemented simulated annealing can be much worst (i.e. slower, and with larger error margins) than a correctly carried out heuristic method. However, it remarked its particular effectiveness in helping to solve topological domain questions when the essence of each problem is cleverly reflected in its structures (valid solutions, permutation sets, cost function and cooling schedule in particular).

In the work by Atkinson [26] additional details of simulated annealing were presented from a purely statistical point of view. Practical implications of limiting the range of random configurations, specifically in segmenting the iteration pool, were followed to two main conclusions. First, when the iterations are evenly segmented over the function domain, it is reasonable to expect one or more segments to converge in less than the allowed maximum number of iterations. Second, to increase the quality of the solution for those segments that converge "early", their search can be re-started from a random solution inside its segment's domain. The paper concluded by reminding the reader that simulated annealing is not universally applicable, that it will not find an optimal solution when the function to evaluate has extreme minima, and that it does not offer any warranty of convergence (not even the segmented version presented in the document).

To preserve the theoretical analysis of the simultaneous multi objective version of simulated annealing, in contrast to multiple applications of simulated annealing for individual independent variables (which weaken the validity of the analysis), Smith et al. [227] defined a mapping of the competing independent variables to a new common domain in which all the combinations of the dependent variables (i.e. the objective functions) were ranked with the use of mathematical dominance. Dominance in this context defines the abstract concept of quality when comparing two solution combinations, and therefore reflects the trade-off relationships of the multiple competing independent variables. Paraphrasing the original authors, "comparing two solution combinations ($\boldsymbol{f}$ and $\boldsymbol{g}$), it is defined that $\boldsymbol{f}$ dominates $\boldsymbol{g}$, if $\boldsymbol{f}$ is no worse than $\boldsymbol{g}$ for all objective functions and it is wholly better for at least one objective function". In essence, a domain transformation of all independent variables from the parameter space into a single objective space.

Since simulated annealing requires a perturbation for iteration (random jump to another possible solution), and since the Pareto front of more than one independent variable defines complicated interactions (e.g. curves, surfaces, etc), the authors included the construction of the attainment surface, evaluation of the energy difference between consecutive solutions, and the verification of the non-dominance of the new random solution by the existing collection of already accepted possible solutions. To promote even coverage of as many parts of the front as possible, the independent variables were modified at random one at a time (from the previous accepted solution), and running statistics kept about the magnitude of all perturbations from which diminishing rescaling factors for the entire set of variables were calculated after each iteration (after a minimum number of iterations to make the statistics more relevant). The application of the rescaling factors to the generation of the next perturbation promoted the convergence of the annealer to solutions closer to the true front than a simple random resampling. The temperature schedule was defined by an initial burn period of algorithm execution, and the initial temperature set to the value where 50% of the proposed solutions were accepted.

A.2.1 Implementation.

Because its relative simplicity to integrate multicriteria decisions, this work conducted a direct implementation of the Rutenbar [218] description of the method with the segmentation improvements suggested by Atkinson [26]. Algorithm A.2 shows the implementation conducted in this work. See subsection A.2 for more details.

Please note that all the terms in algorithm A.2 written in bold type represent multi dimensional structures that were implemented as relational tables in the RDBMS and not as RAM data structures.

The cost function in the algorithm was implemented as a simple arithmetic addition of weighted factors, in which the individual weight multipliers reflected the ratio of relative

Algorithm A.2 Simulated annealing. Based on [218] and [26].

```
Input: $\boldsymbol{Solution\_Set}, \boldsymbol{Perturbation\_SetMin\_T}, \boldsymbol{Max\_T}, Max\_Segments, Burn\_Iterations, Max\_Iterations$
Output: $\boldsymbol{Solution\_Set}, \boldsymbol{Min\_T}, \boldsymbol{Max\_T}$
 1: $Current\_Segment \leftarrow 1$
 2: $Max\_Segment\_Iterations \leftarrow Max\_Iterations/Max\_Segments$
 3: while ($Current\_Segment \leq Max\_Segments$) do
 4:     $Current\_Solution \leftarrow select\_random\_solution(\boldsymbol{Solution\_Set})$
 5:     $Current\_T \leftarrow calc\_solution\_T(Current\_Solution)$
 6:     $Current\_Iteration \leftarrow 1$
 7:     while ($Current\_Iteration \leq Burn\_Iterations$) do                ▷ Burn period
 8:         $Current\_Perturbation \leftarrow select\_random\_perturbation(\boldsymbol{Perturbation\_Set})$
 9:         $New\_Solution \leftarrow calc\_cost\_function(Current\_Solution, Current\_Perturbation)$
10:         $New\_T \leftarrow calc\_solution\_T(New\_Solution)$
11:         if ($| New\_T - Current\_T | \leq \boldsymbol{Max\_T}[Current\_Segment]$) then
12:             $\boldsymbol{Solution\_Set} \leftarrow \boldsymbol{Solution\_Set} + New\_Solution$
13:             $Current\_Solution \leftarrow New\_Solution$
14:             $Current\_T \leftarrow New\_T$
15:         end if
16:
17:         $Current\_Iteration \leftarrow Current\_Iteration + 1$
18:     end while
19:     $\boldsymbol{Max\_T}[Current\_Segment] \leftarrow T$ at which 50% of solutions were accepted.
20:     $\boldsymbol{Cooling\_Schedule}[Current\_Segment] \leftarrow calc\_cooling\_schedule(Current\_Segment, \boldsymbol{Min\_T}, \boldsymbol{Max\_T})$
21:     $\boldsymbol{Cooling\_Schedule\_Transition}[Current\_Segment] \leftarrow calc\_cooling\_transitions(Max\_Segment\_Iterations)$
22:
23:     $Current\_Solution \leftarrow select\_random\_solution(\boldsymbol{Solution\_Set})$        ▷ Search period
24:     $Current\_T \leftarrow calc\_solution\_T(Current\_Solution)$
25:     $Current\_Cooling\_Index \leftarrow 1$
26:     $Current\_Iteration \leftarrow 1$
27:     while ($Current\_Iteration \leq Max\_Segment\_Iterations$) do
28:         $Current\_Perturbation \leftarrow select\_random\_perturbation(\boldsymbol{Perturbation\_Set})$
29:         $New\_Solution \leftarrow calc\_cost\_function(Current\_Solution, Current\_Perturbation)$
30:         $New\_T \leftarrow calc\_solution\_T(New\_Solution)$
31:         if ($| New\_T - Current\_T | \leq \boldsymbol{Cooling\_Schedule}[Current\_Segment, Current\_Cooling\_Index]$) then
32:             $\boldsymbol{Solution\_Set} \leftarrow \boldsymbol{Solution\_Set} + New\_Solution$
33:             $Current\_Solution \leftarrow New\_Solution$
34:             $Current\_T \leftarrow New\_T$
35:         end if
36:
37:         $Current\_Iteration \leftarrow Current\_Iteration + 1$
38:         if ($Current\_Iteration = \boldsymbol{Cooling\_Schedule\_Transition}[Current\_Segment, Current\_Cooling\_Index]$) then
39:             $Current\_Cooling\_Index \leftarrow Current\_Cooling\_Index + 1$
40:         end if
41:     end while
42:
43:     $Current\_Segment \leftarrow Current\_Segment + 1$
44: end while
```

abundance of the independent variable value (e.g. the gender) against the total size of the calibration dataset for all values of that independent variable.

The initial ***Solution_Set*** in the implementation was calculated by taking 101 unique random samples (of each subsequence length, as a starting point suggested in `http://arep.med.harvard.edu/biclustering/yeast.matrix`) and their corresponding solutions were ranked according to their cost function values. The minimum and maximum temperature vectors, per segment, were calculated by sorting the cost function values of the initial random solutions and selecting their minimum and maximum arithmetic values.

In preparation to calculate the ***Perturbation_Set***, the domain of each independent variable was determined from the calibration dataset, i.e. their valid values were determined. To calculate the ***Perturbation_Set***, one valid solution was chosen at random, one of the independent variables was selected in circular fashion and a random perturbation to that value was generated from its domain of valid values. To avoid duplication, the new solution was searched against the ***Solution_Set*** and only new solutions were further considered. If accepted, only the perturbation was added to the ***Perturbation_Set*** because this part of the algorithm does not require a quality check for the new solution.

The ***Cooling_Schedule*** in the algorithm was calculated by successive binary subdivision of the temperature range of each segment, into 10 subdivisions (rounded to the nearest integer and hard-coded in function *calc_cooling_schedule*). For example, if the maximum and minimum temperatures of a given segment were 3,000 and 10 respectively, the *calc_cooling_schedule* would return a vector containing the values {3000, 1505, 758, 384, 197, 103, 57, 33, 22, 16, 10}. Similarly, the number of iterations applied to each subsegment was evenly divided into 10 subdivisions, as hard-coded in function *calc_cooling_transitions* (rounded to the nearest integer as well). Please note that the contents of vector ***Cooling_Schedule_Transition*** is not the amount of iterations to use for every subsegment, it is rather the exact number at which the iteration counter should change the associated temperature value. To finish this example, if $Max\_Segment\_Iterations$ was calculated to be 1,000, the *calc_cooling_transitions* would return vector {101, 201, 301, 401, 501, 601, 701, 801, 901}.

When the calibration dataset was initially used to test the method with the annual and monthly segmentations (subsection 3.6.1.1 and 3.6.1.2), the algorithm did not converge.

To test the correctness of the implementation, two standard datasets were used [*] [†]. Minor problems with the logical operations for the cooling schedule transitions were corrected, until the results of the execution were as expected for these standard datasets. The corrected algorithm was tested once again with the calibration dataset without any improvement in convergence.

A.2.2 Results.

As stated in section A.2.1, the fully implemented method was validated with two independent datasets. But when it was applied to all five versions of the weekly and circadian segmentations of the calibration dataset, combined with the two log-likelihood ranking measures, convergence was not obtained with any combination of cooling schedule (ten different alternatives tested) and iteration budget (1,000, 2,000, 5,000 and 10,000), revealing the highly cyclic nature of this dataset. In operational terms, the quality of each successive solution given by the cost function never improved as expected by the structure of the cooling schedule, no matter how many iterations were allowed for this dataset.

This is an important limitation for the application of the simulated annealing method for the tracking dataset.

Alternatively, this lack of convergence could be due to a major conceptual mistake. Most probably in the calculation of the valid solutions, the permutation sets, and/or the structure and complexity of the cost function. To detect and correct any problems of this nature, the researcher needs a considerable amount of additional experience in the application of this tool.

As a consequence, this method was also deemed unsuitable.

[*] `http://arep.med.harvard.edu/biclustering/yeast.matrix` and
[†] `http://genome-www.stanford.edu/scleroderma/data.shtml`

Appendix B

Generalizing moving objects and patterns.

Beyond specific solutions to special cases, some scientists have started to ask questions about the fundamental nature of moving objects and the associated datasets used to capture those movements. Their main objective is to create a standard framework to formally define those fundamental aspects of moving objects that will serve to better classify and study them.

At this time, there is no definitive consensus on what are the best metrics for the study of moving objects or Behavior Monitoring and Interpretation (BMI). As described in section 2.3, this difficulty is due in part to the variety of specific aspects that each study focuses on. However, the existence of a solid framework for the study of moving objects would allow scientists to quickly classify their special cases and then identify the most appropriate analysis techniques. Additionally, the fundamental elements of movement can be used to program specific behaviors in specialized applications (e.g. smart home environments, mobile robots, smart mobile agents in geosimulations) making a natural language interface available for the end users.

Directly translating the time saved by the use of the framework into a monetary benefit is important, but the additional benefit would be to free researchers from minor aspects of their studies, to let them concentrate on more challenging ones. The motivation to find a standard framework to describe moving objects go beyond the purely academic domain. Once in place, explaining the complex behaviors of moving objects to the general public will be easier, and in time, the associated technological advances could be adopted in a similar way as the Geographic Information System (GIS)-derived technologies have been assimilated in the past decade.

The objective of this appendix is to include some recent research efforts towards the creation of the framework, to use it in this work and expand its definition if possible.

Current scientific studies towards the creation of the framework can be roughly divided according to their focus, purely topological studies are included in section B.1, studies that integrate topological and behavioral concepts are presented in section B.2, studies concentrating on trajectories and patterns for groups of moving objects are discussed in section B.3, and other approaches are briefly described in section B.4.

B.1 Topological perspective.

An early publication by Egenhofer [84] describes, in mathematical detail, the possible relationships amongst geometric and geographic entities. The document makes an explicit definition of the two-dimensional space where the entities exist, keeping it generalized at the same time to accept any higher dimensionality. Four types of relationships are identified, *topological, spatial order, metric,* and *fuzzy.* The *topological relationships* are defined by the characteristics of their parts (i.e. continuity, closure, interior and boundary), and their invariability under topological transformations (i.e. translation, rotation and scaling). The *spatial order* relationships are defined in terms of an ordering of the entire representation space, making them directly dependent on this order (e.g. *behind* and *in_front_of*). The *metric relationships* are defined as geometric consequences of the use of specific distances and operational values, (e.g. all towns 200 km or less from downtown Calgary). *Fuzzy relationships* are defined as consequences of movement of one or more objects (e.g. *through* and *into*), or as loose geometrical consequences that characterize a qualitative condition given by the user of the system (e.g. all municipalities *near* Kananaskis Country).

Concentrating on the topological relationships amongst intervals of values in one-dimensional space[*] and simple regions[†] in two-dimensional space, the paper defines a set of eight relationships according to the intersection of their boundaries and interiors (*disjoint, meet, overlap, inside, contains, covers, coveredBy* and *equal*). A generalization of the initial concepts was presented to extend their application to the three-dimensional space. This approach was also implemented using a relational database engine as proof of the viability of the analysis [89].

Later publications by Egenhofer et al. analyzed additional details of the formal mathematical expressions of topological relationships, topological interactions of objects of different dimensionality (e.g. a line interacting with a region), and practical considerations for the implementation of an expanded query language that would include topological operations (e.g. [85, 88, 90, 95]). Relevant to this period, publication [94] expanded the analysis of the topological relationships to include the exteriors of the elements, and proposed the *9-intersection* matrix (equation B.1) as a way of summarizing the topological interactions between any pair of simple regions.

$$R(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \tag{B.1}$$

The *9-intersection* matrix for a pair of simple regions, A and B, defines the topological *intersections* of their respective *interior* ($^\circ$), *boundary* (∂), and *exterior* or complement ($^-$). Each one of the eight relationships (*disjoint, meet, overlap, inside, contains, covers, coveredBy* and *equal*) produces a different pattern in the corresponding *9-intersection* matrix.

Direct comparison between the *9-intersection* matrix and its predecessor, the *4-*

[*]Simple lines: Lines with only one starting point and one ending point (no branches), and that do not self-intersect.

[†]Fully connected regions without holes, with a continuous boundary, and that do not self-intersect.

intersection matrix [84, 90, 95], was studied in publication [98]. The main difference between both representations being that, the *9-intersection* matrix includes specific intersections of the exterior of the elements, while the predecessor does not. Therefore, the *4-intersection* matrix has important limitations when analyzing topological relationships in two-dimensional space, however, these limitations do not exist when analyzing elements of co-dimension 0 (i.e. lines with lines, and regions with regions). It is not until both matrices are used to analyze topological relationships of elements of co-dimension 1 (i.e. lines with regions), that the limitations of the *4-intersection* matrix are truly obvious. As a consequence, the *9-intersection* matrix encapsulates more detail about topological relations. However, when implementing spatial queries for GIS applications not all the cells in the *9-intersection* matrix have to be calculated every time, before the overall relationship between two objects can be characterized, and an answer generated for the user.

Subsequent work contributed additional theorems and proofs to the formal definition of the relationships, some of their properties (i.e. symmetry, composition, uniqueness, transitivity, completeness, non-orthogonality), characteristics and definitions of the topological relationships of complex regions (i.e. regions with holes or regions composed of disconnected areas), metric constraints, implementation guidelines, computational implementation improvements, analysis of the equivalent relations in spherical space, etc. (e.g. [54, 55, 59, 60, 61, 86, 91, 92, 93, 96, 97, 105, 106, 135, 136, 149, 186, 202, 214, 215, 241]).

To close the gap between purely conceptual analysis and practical applications, some projects have also studied the topological relationships between trajectories (directed lines) and regions of different types in two-dimensional space. The publication by Kurata and Egenhofer [153], proposed the *9^+-intersection* matrix (equation B.2), an enhanced version of the original, to represent relationships between simple regions and the starting

($_s$) and ending ($_e$) points of L_D.

$$M^+(L_D, R) = \begin{bmatrix} L_D^\circ \cap R^\circ & L_D^\circ \cap \partial R & L_D^\circ \cap R^- \\ \begin{pmatrix} \partial_s L_D \cap R^\circ \\ \partial_e L_D \cap R^\circ \end{pmatrix} & \begin{pmatrix} \partial_s L_D \cap \partial R \\ \partial_e L_D \cap \partial R \end{pmatrix} & \begin{pmatrix} \partial_s L_D \cap R^- \\ \partial_e L_D \cap R^- \end{pmatrix} \\ L_D^- \cap R^\circ & L_D^- \cap \partial R & L_D^- \cap R^- \end{bmatrix} \tag{B.2}$$

Additional studies with this approach expanded the model with complementary theoretical treatments, metric constraints, line-line interactions (directed lines), neighborhood considerations, schematic representations, ontological characteristics, etc. (e.g. [42, 62, 87, 151, 152, 154, 208, 232]).

Also from a topological theoretical point of view, but with the objective to generate qualitative language descriptions for the control of moving agents (e.g. mobile robots, multi-agent simulations), the paper by Dylla [83] presented a general description of the concepts of QSR and how to apply them for control of agents.

The paper started by explaining the different challenges involved in a qualitative spatial representation of moving entities, namely, formal qualitative quantifications of orientation, location, distance, motion and change in those quantifiers. i.e. how to formally capture "everyday commonsense knowledge about space with a limited amount of symbols ... without numerical values". Next, qualitative spatial reasoning was considered from a mathematical point of view to define formal inference mechanisms known as qualitative spatial calculi. Options of which include analysis of operations on topological relations, constraint-based, and neighborhood-based reasoning and their respective distinctive requirements (i.e. operands and operations, consistency, and action description).

Considering calculi to represent relative orientation of moving objects, the work analyzed the advantages and limitations of the flip-flop, single/double cross and dipole calculi, as well as the strengths and shortcomings of the Dipole Relation Algebra (DRA) and two versions of the oriented point relation algebra (i.e. $OPRA_m$ and $OPRA_m^*$) [83].

In the last section of the study, the author considered a hybrid implementation of the presented mechanisms for the control of moving agents. Complex cases for the combination labeled "action-augmented conceptual neighborhood graph (ACNg) for $OPRA_m^*$" were explored for a single rotating object, simultaneously rotating objects and circular motion description and analysis. To close the chapter, the developed concepts were used to implement a moving agent control framework for maritime navigation (composed of the qualitative scene description, the qualitative rule representation and the symbolic reasoning modules), employing a stepwise approach (i.e. an iterative, approximate and refine development), and a back propagation loop between the environment, the qualitative framework, and the action primitives.

The product of the study, the Rule Transition System (RTS), was identified in the general conclusion of the document as an specialized framework version applicable to vessel navigation (i.e. an specialized implementation of the framework). The generalization of this framework made intrinsically possible by the individual applicability and presentation of its separate parts.

B.2 Combining topology and behavior.

Integration of topological concepts and behavioral interpretation can be found in [155]. The analysis started with the identification of the 19 topological relationships between simple lines (non-directed) and simple regions in two-dimensional space using the *9-intersection* matrix (equation B.1). The conceptual neighborhood graph of these relationships was presented as a two-dimensional organizational device to understand the possible transitions from one relationship to another, i.e. how a single change in one of the *9-intersection* matrix' cells will transform that pattern into another connected neighboring pattern.

Using the *9^+-intersection* matrix (equation B.2) for trajectory analysis, 26 unique topological relations were identified between simple regions and directed lines in two-dimensional space. In contrast with the case for non-directed lines, the conceptual neighborhood graph for these relations can be represented in different ways. The authors present three alternatives, as a three-dimensional structure, as two bi-dimensional graphs that share 12 peripheral patterns, or as the authors prefer, as an iconic structure with partitioned nodes to represent the upper and lower halves of the three-dimensional equivalent. Combining the neighborhood graph concept with the classification schema proposed by Gao et al. [111], the researchers developed the iconic representations of groups of topological relations that satisfy qualitative conditions (e.g. start from the inside, start from the border, start from the outside).

Using the *IBE-sequence* notation from [154], the authors codified the 26 topological relations as an implicit three part regular grammar *(S, T, N)*[*] [132, 134, 140, 141, 175, 226], consisting of the intersection of the start point of the directed line with the region *(S)*, a sequence of transitions of the intersection of the interior of the directed line between interior, boundary and exterior of the region *(T)*, and the intersection of the end point of the directed line with the region *(N)*. Defined here as the Start-Transition-eNd (STN) notation. Table B.1 shows the terminal symbols of the equivalent regular grammar, table B.2 contains the generic not-normalized production rules, and table B.3 shows the valid combinations of the production rules that represent the topological interactions between the directed line and the region.

After the identification of the unique topological relations, twelve of them were found to be direction-invariant, while the remaining fourteen were described as direction-variant

[*]Modified from the original author's *W, X, Y* and *Z* notation for the more concise presentation of the production rules included here.

Table B.1: STN terminal symbols.

Symbol	Interpretation
I	Intersection of the directed line with the *interior* of the region
B	Intersection of the directed line with the *boundary* of the region
E	Intersection of the directed line with the *exterior* of the region

Table B.2: STN production rules.

S	T	N
$S_1 \rightarrow \mathrm{I}$	$T_1 \rightarrow \mathrm{I}$	$N_1 \rightarrow \mathrm{I}$
$S_2 \rightarrow \mathrm{B}$	$T_2 \rightarrow \mathrm{B}$	$N_2 \rightarrow \mathrm{B}$
$S_3 \rightarrow \mathrm{E}$	$T_3 \rightarrow \mathrm{E}$	$N_3 \rightarrow \mathrm{E}$
	$T_4 \rightarrow T_1 \mid \emptyset$	
	$T_5 \rightarrow T_2 \mid \emptyset$	
	$T_6 \rightarrow T_3 \mid \emptyset$	
	$T_7 \rightarrow T_1\ T_2 \mid T_1\ T_2\ T_7$	
	$T_8 \rightarrow T_2\ T_1 \mid T_2\ T_1\ T_8$	
	$T_9 \rightarrow T_3\ T_2 \mid T_3\ T_2\ T_9$	
	$T_{10} \rightarrow T_2\ T_3 \mid T_2\ T_3\ T_{10}$	
	$T_{11} \rightarrow T_1\ T_2\ T_3 \mid T_3\ T_2\ T_1$	
	$T_{12} \rightarrow T_7 \mid T_9$	
	$T_{13} \rightarrow T_8 \mid T_{10}$	
	$T_{14} \rightarrow T_{12} \mid T_{12}\ T_{14} \mid \emptyset$	
	$T_{15} \rightarrow T_{13} \mid T_{13}\ T_{15} \mid \emptyset$	
	$T_{16} \rightarrow T_5\ T_7 \mid T_8\ T_5$	
	$T_{17} \rightarrow T_5\ T_9 \mid T_{10}\ T_5$	

relations.

Moving objects are usually observed for a relatively long period of time, and their resulting trajectories describe complicated motions. To efficiently detect the actions of the moving objects in relation to regions, i.e. to detect the STN sequences and the equivalent of the iconic groups of topological relations, the problem can be decomposed into the analysis of relatively small sub-trajectories and individual regions. The next section of the original study briefly considered five segmentation strategies; equal-length, equal-interval, boundary-based, primitive-based and temporally-adjusted primitive-based. The fourth and fifth segmentation strategies are the preferred method for computational purposes, since their definitions allow the assignment of priorities for detection, and in consequence,

Table B.3: STN representation of topological interactions between a directed line and a simple region, based on [155].

Direction-Invariant relations			
	S	T	N
a	S_1	T_1	N_1
b	S_1	$T_7\ T_1$	N_1
c	S_1	$T_1\ T_2\ T_{14}\ T_3\ T_{15}\ T_2\ T_1$	N_1
d	S_3	$T_3\ T_2\ T_{14}\ T_1\ T_{15}\ T_2\ T_3$	N_3
e	S_3	$T_9\ T_3$	N_3
f	S_3	T_3	N_3
g	S_2	T_1	N_2
h	S_2	T_{16}	N_2
i	S_2	$T_5\ T_{14}\ T_{11}\ T_{15}\ T_5$	N_2
j	S_2	T_{17}	N_2
k	S_2	T_3	N_2
l	S_2	T_2	N_2

Direction-Variant relations			
	S	T	N
m_1	S_1	T_1	N_2
m_2	S_2	T_1	N_1
n_1	S_1	$T_7\ T_4$	N_2
n_2	S_2	$T_4\ T_8$	N_1
o_1	S_1	$T_1\ T_2\ T_{14}\ T_3\ T_{15}\ T_5$	N_2
o_2	S_2	$T_5\ T_{14}\ T_3\ T_{15}\ T_2\ T_1$	N_1
p_1	S_1	$T_1\ T_2\ T_{14}\ T_3$	N_3
p_2	S_3	$T_3\ T_{15}\ T_2\ T_1$	N_1
q_1	S_2	$T_5\ T_{14}\ T_1\ T_{15}\ T_2\ T_3$	N_3
q_2	S_3	$T_3\ T_2\ T_{14}\ T_1\ T_{15}\ T_5$	N_2
r_1	S_2	$T_6\ T_{10}$	N_3
r_2	S_3	$T_9\ T_6$	N_2
s_1	S_2	T_3	N_3
s_2	S_3	T_3	N_2

for implementation.

An implicit restriction in the grammar presented above (included in the production rules), does not allow the contiguous repetition of the same terminal symbol in the T rules. If repetition was allowed, or generated by any of the forms of segmentation described in the preceding paragraph, there would be no change in the interaction between the directed line and the region. Therefore, that repetition would not contribute any new topological information to the analysis. If this kind of repetition was present in the dataset, it would have to be eliminated from the segmentation itself or in one of the subsequent preprocessing steps.

To expand the application of the 9^+*-intersection* matrix to three-dimensional space, the authors considered the 45 topological relations between a two-dimensional simple region and a three-dimensional directed line (first presented in [151]). Of this group of relations, 26 are equivalent to those in the purely two-dimensional case, and only 19 relations are new in the analysis. As the authors pointed out, the largest limitation of their topological analysis can be seen when the directed line does not interact with the region at all, however, they also suggested complementary analysis strategies [224, 225] that can account for this situation.

In their conclusion, the authors considered the additional tasks to fully analyze the interactions between a three-dimensional directed line and a two-dimensional region. Some of those tasks include the development of a new iconic representation of neighboring topological relations, the identification of a set of primitive motion patterns to detect motion intentions, and the definition of the new topological relations in the *IBE-sequence* notation.

B.3 Trajectory and pattern perspective.

In the study by Laube and Imfeld [159], relative motion of a group of moving objects was analyzed in two dimensional space. To find common essential information, the trajectories for individual moving objects were transformed into a conceptual space that represents motion azimuth, speed, and change of speed. To simplify the analysis, an absolute time origin was implied by characterizing the positions as being observed synchronously. The RElative MOtion (REMO) analysis matrix directly detects patterns over time, patterns across objects and combined patterns (patterns over time and across objects). But additional information is needed to differentiate between patterns amongst objects in close proximity (related) and patterns detected between objects that are separated by a long distance (not related).

Further refinements in the definitions of the techniques and patterns can be found in Laube et al. [161]. Different measurements of proximity amongst moving objects were considered, and the computational cost of the algorithms to solve the *flock, leadership, convergence* and *encounter* patterns was given.

Gudmundsson et al. [120] studied the same four movement patterns from [161], still with the synchronous time constraint, and proposed algorithms that find their approximate solutions (faster than their respective exact solutions). Table 1 in the original publication ([120], pg. 251) shows the computational costs for the four patterns in both studies ([161] and [120]). The following definitions of the patterns were taken from the original text ([120], pg. 251).

"**Flock** Parameters: $m > 1$ and $r > 0$. At least m entities are within a circular region of radius r and they move in the same direction.

Leadership Parameters: $m > 1$, $r > 0$, and $\tau > 0$. At least m entities are within a circular region of radius r, they move in the same direction, and at least one of the entities was already heading in this direction for at least τ time steps.

Convergence Parameters: $m > 1$ and $r > 0$. At least m entities will pass through the same circular region of radius r (assuming they keep their direction).

Encounter Parameters: $m > 1$ and $r > 0$. At least m entities will be simultaneously inside the same circular region of radius r (assuming they keep their speed and direction)."

Figure B.1 shows these patterns in a three dimensional representation with the vertical axis displaying the time domain.

In the chapter by Laube et al. [160] on analysis of point motion, the REMO transformation (including the necessary geographic constraints and the synchronous time constraint) was proposed as one of the data mining methods for knowledge discovery techniques. The authors emphasized the need for experts in the field of the subjects being studied through their movement datasets (e.g. experts on grizzly bear behavior). These experts should define the best way of quantifying the relevance of the patterns in the data mining process and interpret their meaning. The study also reminded the reader that scale issues have to be considered carefully to avoid the modifiable areal unit prob-

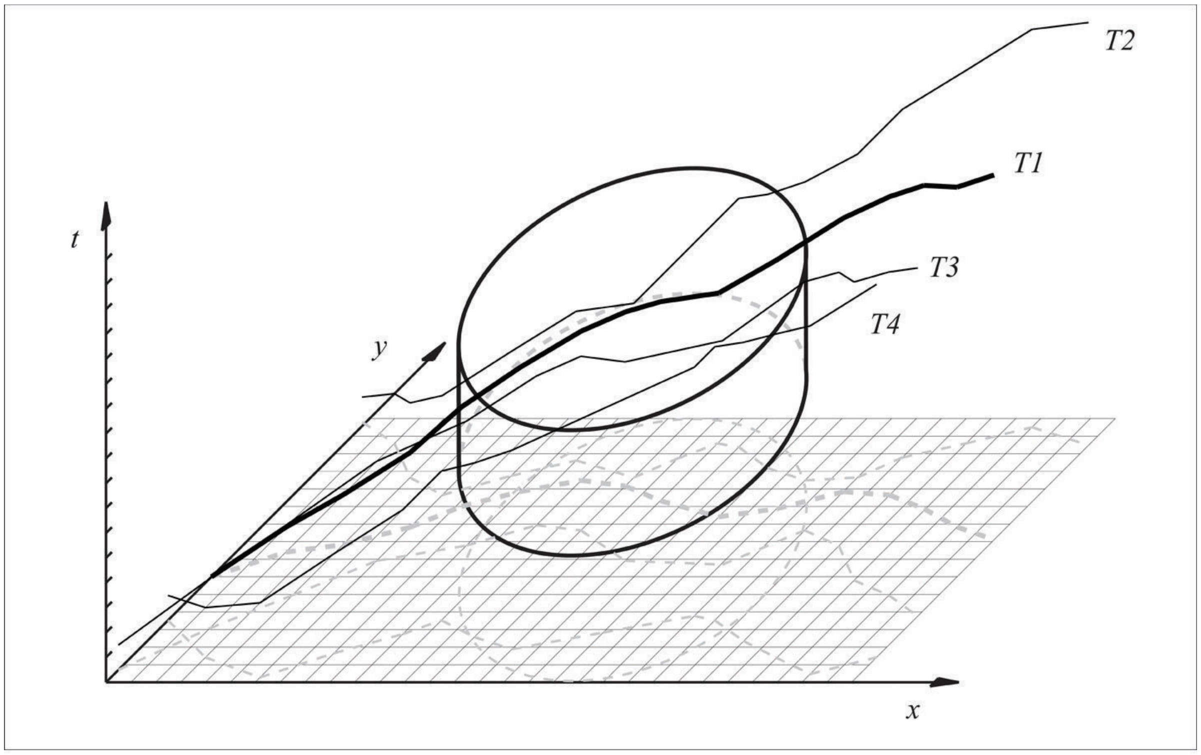

Figure B.1: Generic movement patterns of a group of moving points. Combining the concepts by [81], [120], [159] and [161]

lem, and its newly minted equivalent in the time domain, the modifiable temporal unit problem [160].

As a departure from the REMO framework, but still in two-dimensional space, Andersson et al. [11] re-examined the basic definition of the *leadership* pattern to make it more flexible for application in studies of wildlife movement. The paper defined the pattern in terms of the minimum number of moving objects that have to take part in it (m), the radius of the influence region of each moving object (r), an apex angle for each moving object (α), and the absolute angle between the heading vectors of every pair of moving objects (β). Figure B.2 shows these measurements between two moving objects, except for m and β that are abstract user-defined parameters.

In the figure, e_j is in front of e_i, and both are heading towards d_j and d_i, respectively. If $\|d_i - d_j\| \leq \beta$ then e_i follows e_j.

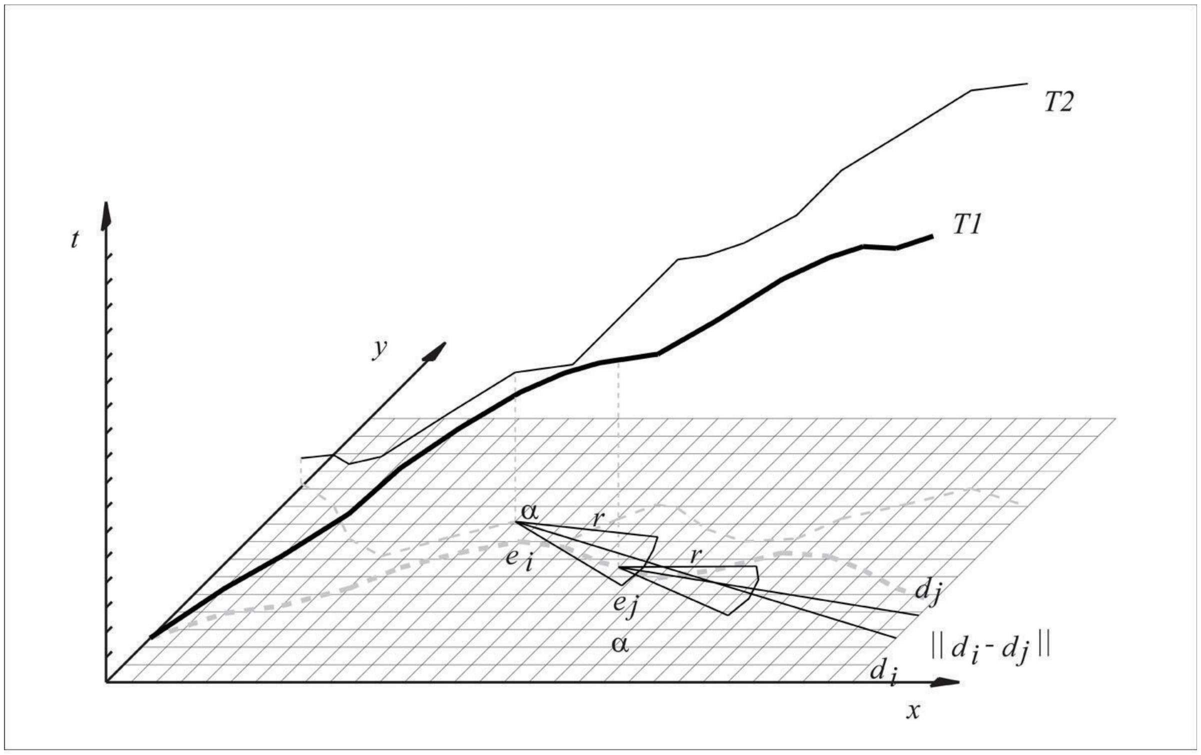

Figure B.2: *Leadership* pattern as a function of the geometric relationships between two moving objects. Adapted from [11] where e_j is in front of e_i. If $\|d_i - d_j\| \leq \beta$ then e_i follows e_j.

Since the values for m, r, α and β depend on the species being studied, the authors only proposed generic algorithms to solve the pattern. The study divided the identification of the *leadership* pattern into three related special cases:

- All time intervals where a single moving object is the leader of at least m other entities for at least k time units.
- The longest time period (k^{max}) where the same moving object is the leader of at least m other entities.
- The largest number of entities (m^{max}) where the same moving object is their leader for at least k time units.

Each special case has four different variations according to the combinations of their discrete and continuous version and the consistency[*] or variation[†] of the group of followers. Analysis included the generic solution for all cases and provided their computational complexity cost.

The study by Andrienko and Andrienko [12] presented an integration effort towards the definition of a unified framework for the analysis of moving objects. In this study their emphasis was the analysis of large collections of movement data and the creation of visual tools to assist in the research effort. First, they identified the primitive characteristics of movement data in their environment as:

1. Properties of space (e.g. altitude, slope, obstacles).
2. Properties of time (e.g. cyclic, intensity, working day vs. weekend).
3. Properties and activities of the moving entities (e.g. age, gender, means of transportation).
4. Various spatial and spatio-temporal phenomena (e.g. climate, weather).

Their next step was to describe movement patterns in generic behavioral terms as:

- IMB. Individual movement behavior
- MCB. Momentary collective behavior.
- DCB. Dynamic collective behavior.

The Dynamic Collective Behaviors (DCBs) can be decomposed either as similar Individual Movement Behaviors (IMBs) for the entire set of entities or, as similar Momentary Collective Behaviors (MCBs) at all time moments. Seeing how the IMBs and MCBs are the essential expressions of behavioral patterns, the researchers defined the similarity

[*]The same moving objects for the entire time period.
[†]Different moving objects but m or m^{max} in number at all times.

measures for each one. Similarity between IMBs means similarity in overall characteristics, co-location, synchronization in time, and co-incidence in space and time. Similarity between MCBs is described over time as constancy, change, fluctuation, pattern change, repetition, periodicity and symmetry. The study continued by providing general rules for segmentation and aggregation of movement data and some transformations in space and time to identify movement characteristics and patterns (IMBs and MCBs). Examination of IMB clustering and visual presentation of the clustering results was considered next. The last part of the paper presented general guidelines to use the analysis and presentation methods developed in their study as interactive and visual techniques. Their conclusion stated that all the above levels of abstraction and aggregation can be used in an integration process to acquire overall knowledge of the underlying phenomena.

The work of Dodge et al. [81] represents a more comprehensive effort towards the definition of a consolidated framework for the classification and analysis of moving objects. From the start, the authors reminded the reader of the significance of using the correct scales, spatial and temporal, according to the pattern and behavior they need to capture. Based partially on the studies presented this far in the appendix, the authors proposed a classification of datasets according to two levels of patterns (generic and behavioral), and three types of movement parameters (spatial, temporal and spatio-temporal). The generic patterns are subdivided into primitive and compound according to the implicit analysis of individual aspects or subsections of the entire movement, while the behavioral patterns are interpretations of the entire dataset mostly left to experts in each specialty (i.e. wildlife experts and behavioral ecologists). The subdivision of the generic patterns continues until one or more fundamental parameters are identified as the primary aspects of the movement. Primitive parameters are position, instance (purely as a function of time) and interval. Primary derivatives are those aspects that can be directly defined as a function of the primitive parameters, such as distance, direction, duration and velocity. Secondary derivatives are defined, in an iterative fashion, as functions of the primary derivatives and include change of direction, change of duration, acceleration and approaching rate amongst others. The paper includes examples of all the proposed generic and behavioral patterns in an effort to clearly illustrate the concepts and application of the proposed framework. Please refer to table 3 in the original publication ([81], pg. 7) where the proposed generic patterns were compared, and as a consequence defined, in terms of the primitive parameters, primary derivatives and secondary derivatives. The details of the definition of the example behavioral patterns, in terms of the proposed framework, can also be found in the original publication ([81], pg. 10). Towards their conclusions, the authors made an open invitation to other experts in the field to contribute their own point of view about this framework in a moderated online forum*.

An additional review of advances on pattern analysis from moving object trajectories by Laube can be found in [158]. The chapter integrated concepts from the previous works presented in this section, with the objective of summarizing the best established theories in the analysis, identification and application of moving object patterns.

The document started by defining five conceptualizations necessary to study moving object patters. First, the abstraction of representing almost all moving objects as moving points, to disregard the complications of their spatial extension and change over time (e.g, storm front evolution, fish school volumetric variations, rush hour car accumulation and spread). Acknowledging the limitations of modern computer platforms to represent and even store continuous movement paths, the paper continued by justifying the conversion of the analog moving point path into a digital equivalent (i.e. the use of the conceptual space-time path constructed from discrete positions in two or three dimensional space with a time dimension). The publication continued by stating the existence of several analytical tools for static scene analysis, and the need of equivalent

*`http://movementpatterns.pbwiki.com/Patterns-of-Movement`

tools that explicitly incorporate the time dimension. After these, the research continued by considering the difficulty of defining universal analysis parameters, specially metric parameters, for the study of all moving pattern. This difficulty being a reflection of the specialized nature of the cases that each method has been applied to, and the additional difficulties brought into play by modifying the method to analyze a different, but similar, case and dataset. The final consideration of this section highlighted the influences, even constraints, that the surrounding environment imposes on the behavior of the moving objects. A consideration that has to be incorporated to some degree into the analysis tools, to enable them to produce positive identification of a set of standard patterns.

The following section examined some of the subjective factors that have to be represented into the proposed solutions, and then arrived at the following definition.

"**Definition 1**. A movement pattern is any high-level description of the movement of an individual or a group of individuals. This description can but must not relate the movement to the underlying space." ([158], p. 48)

The third section of the chapter considered the advantages and disadvantages of six conceptual spaces for the embedding and analysis of movement models. *Euclidean homogeneous space* was defined as an empty environment in $\mathbb{R}^2$, where the space itself does not impose any influence on where the objects can move to. To enumerate the limitations of this representation, the chapter defined *first order effects* as those relations between environment and moving object, and *second order effects* as inter-object relations. The discrete position representations on this space are commonly referred as fixes. The next representation space was the *Constrained Euclidean space*, which is the equivalent to the previous space, but adds physical elements to the environment that restrict the movement of the objects (e.g. highways, ridges, cattle fences and movement corridors in countryside and wildlife studies, buildings in urban environments, walls, partitions and furniture in home environments). The *space-time cube* was defined as a three-dimensional representation of movement with two axis defining an Euclidean space and the third axis representing the time dimension. This representation is popular in time geography and is commonly used for visualization purposes. In this space, movement trajectories take the form of space-time paths. *Heterogeneous field space* are a consequence of study area data being represented as conceptual fields and digital raster forms. This conceptualization is preferred in behavioral ecology studies, where the moving agents have to interact with a heterogeneous environment to determine their next location (most of the time based on neighborhood resources and cost evaluation processes). *Irregular tessellations* are routinely used in LBS studies [150], where these can be constructed from cellular tower signal strength maps or more commonly, from Voronoi tessellations of the study area. *Network space* is mainly used in urban transportation studies, with street and train track representations in the form of graphs (directed or not). In recent studies, the initial homogeneity of travel speeds has being adjusted to incorporate velocity vector fields and cost surfaces (e.g. [178]). Each one of there space representations being most advantageous for analysis with specific goals, and each one being susceptible to topological and operational transformation from one type to another.

In the following section of the document, Laube identified "signature patterns" using the analysis structure proposed by Han and Kamber [128]. In broad strokes, summarized in table 2 of the original publication ([158], pg. 60): Description of patterns was achieved in terms of summary quantities (i.e. speed, sinuosity, average daily velocity, average daily direction of travel) that tend to reflect a static perspective not suitable for dynamic conditions (depending on the temporal resolution). The identification of frequent patterns promises qualitative descriptions with the advantages of intuitive interpretations, but thus far does not have statistical relevance measures or solid scientific application [158]. The classification and prediction methods offers the possibilities of simulations and control of moving agents, but its largest limiting factor is the considerable amount of context knowledge needed for an efficient implementation. Finally, the clustering of trajectories (or sub-trajectories) based on similarity measures offers the identification of geographically referenced regions where specific types of trajectories abound, however, its largest limiting factor as an exploratory method prevents it from being used beyond a visual exploration tool [158].

The fifth section of this chapter included detailed descriptions of the specialty disciplines in which each type of moving pattern analysis can be used to understand subjacent phenomena. The main applications described include prerequisite and implementation considerations for exploratory data analysis, visualization, moving object databases, surveillance imagery analysis, and behavioral ecology studies. Section six in the chapter offered arguments to identify the main limitations for the primitive elements considered for movement pattern analysis, such as the pattern definitions, first order effects, second order effects, pattern relevance measurements, scale and granularity repercussions. The seventh section of the chapter considered some of the improvements that can be achieved in this specialty with future research work, as well as some of the practical solutions that can motivate the improvement of these tools (e.g. real time traffic jam avoidance, automated assisted living environments).

In his conclusion, the author reminds the specialists in this field that the underlying theory is still incomplete and unproven for commercial and social value.

B.4 Other perspectives.

In the same volume as the previous document, the paper by Wood and Galton [246] presented a theoretical review of most of the work included in this appendix, plus [237]. The authors analyze all aspects of the previous solutions with particular care of the undefined portions of the theoretical framework as it pertains to collective movement, and then presented their own qualitative framework for the analysis of large volumes of moving object data. In their conclusion, the chapter acknowledged the preceding fundamental concepts and methods to describe and analyze moving object data, and the further need for projects looking to explain and understand collective movement.

From a slightly different perspective, time geographers have been trying to define a standard analysis framework for movement in the space-time domain. The paper by Miller [177], amongst other constructs, showed the characteristics of the space-time path and prism, which are remarkably similar to the definitions of trajectories and regions of the rest of the papers cited in this appendix, but with explicit treatment for the time dimension. This relatively early approach is focused to the analysis of human movements constrained to a urban transportation network. However, in a later work by Miller and Bridwell [178], those constraints were generalized by the use of velocity fields, and the study of potential path areas and space-time prisms. Their objective was to define a theoretical model that can be applied to the prediction of possible movement paths for different human activities. The final product was a generalized theory to model interactions between objects (in the traditional time geography sense and as network time prisms) and fields (that describe intensities of time geographic properties for specific locations). As the authors point out, the later unification of both perspectives can start with the formal theoretical definition of wildlife movement in terms of the equivalent objects (e.g. individual GPS locations, individual bear trajectories) and fields (e.g. accumulated cost surfaces).

From yet another point of view, as detailed in section 2.4, the main task of identifying patterns in a set of trajectories can be seen as finding the longest common subsequences amongst trajectories.

Appendix C

Calibration dataset and tables of Grizzly bear tracking data.

This appendix includes tabular representations of all significant data used in this work derived from the raw GPS point data. Section C.1 contains a description of the calibration dataset, while section C.2 includes the results of the χ^2 test on the cleaned dataset. The tables in section C.3 are simple segmentations of the full dataset based on previous studies and, section C.4 contains a more detailed presentation of the temporal distribution of the GPS data.

C.1 Calibration dataset.

The table in this section contains a detailed description of the GPS points selected as the calibration dataset for this work. The set includes data acquired at different temporal resolutions, during different years and, as much as possible, from different geographical locations inside the study area.

The full dataset, after data cleaning and integration (sections 3.3 and 3.4), includes 199,223 individual GPS positions. The subjective process to select the calibration dataset is detailed in section 3.5 and resulted in 35,850 individual GPS positions, which represents 18% of the larger set. Table C.1 identifies the individual annual grizzly trajectories included in the calibration dataset.

Table C.1: Data selected as the calibration dataset.

Year	Bear	GPS points
1999	G002	835
1999	G003	10
1999	G006	15
1999	G008	22
1999	G016	641
1999	G020	187
2000	G016	56
2000	G023	227
2000	G026	25
2000	G027	741
2000	G028	17
2000	G029	122
2000	G033	126
2001	G004	47
2001	G016	299
2001	G024	903
2001	G029	1,130
2001	G033	189
2001	G036	106
2001	G038	98
2001	G040	26
2001	G100	159
2002	G003	24
2002	G008	165
2002	G012	28
2002	G023	52
2002	G027	91
2002	G028	702
2002	G035	12
2002	G037	375
2002	G040	73
Sum		7,503
2003	G003	463
2003	G010	48
2003	G023	87
2003	G028	679
2003	G040	65
2003	G043	129
2003	G048	74
2003	G061	102
2003	G068	22
2003	G071	152
2003	G073	24
2003	G074	43
2003	G100	60
2003	G106	77
2004	G004	92
2004	G064	20
2004	G065	58
2004	G071	29
2004	G072	79
2004	G073	11
2004	G075F	68
2004	G077	397
2004	G078	27
2004	G080	612
2004	G086	95
2004	G095	33
2004	G099	696
2004	G89K	35
Sum		4,277
2005	G059	49
2005	G087	262
2005	G095	18
2005	G097	81
2005	G201	228
2005	G205	79
2005	G207	159
2005	G208	16
2005	G216	850
2005	G217	10
2005	G218	1,373
2005	G225	13
2005	G226	580
2005	G230	1,172
2006	G040	391
2006	G204	126
2006	G205	143
2006	G218	7,983
2006	G225	33
2006	G230	940
2006	G236	135
2007	G223	2,353
2007	G238	161
2007	G260	526
2007	G262	104
2007	G264	83
2007	G266	1,712
2008	G077	3,160
2008	G110	92
2008	G223	305
2008	G224	139
2008	G230	145
2008	G254	175
2008	G265	461
2008	G266	13
Sum		24,070
	Total	**35,850**

C.2 χ^2 test.

Table C.2 presents the numerical results of the χ^2 test from section 3.4, comparing the land cover occurrence distributions of the full study area against the GPS-land cover matched positions, with the explicit inclusion of the "n/a" class. See table 4.14 in subsection 4.2.4) for the observed land cover values (column "Original").

Table C.2: χ^2 test.

Symbol	Observed (O)	Expected %	Expected (E)	O - E	(O - E)^2	(O - E)^2 / E
0	484	0.3712	739.52	-255.52	65,288.31	88.29
1	105,793	54.3606	108,298.82	-2,505.82	6,279,124.54	57.98
2	3736.0000	6.1725	12,297.04	-8,561.04	73,291,400.32	5,960.08
3	21,693.00	13.5581	27,010.85	-5,317.85	28,279,566.52	1,046.97
4	1,072.00	0.9159	1,824.68	-752.68	566,532.39	310.48
5	44,975.00	8.6798	17,292.16	27,682.84	766,339,743.74	44,317.18
6	325.00	1.7615	3,509.31	-3,184.31	10,139,850.21	2,889.41
7	20,673	12.5657	25,033.76	-4,360.76	19,016,267.12	759.62
8	163	1.1801	2,351.03	-2,188.03	4,787,478.01	2,036.33
9	6	0.0314	62.56	-56.56	3,198.58	51.13
A	303	0.4032	803.27	-500.27	250,267.21	311.56
Total	199,223	100.0000	199,223.00	0.00		**57,829.05**

C.3 Simple segmentations.

After data cleaning and integration (as described in sections 3.3 and 3.4), the data in tables C.3 to C.6 was obtained.

The definition of annual core home range was initially adopted from [238], and was recalculated for the present work with the application by [162], but ultimately replaced by the corresponding Voronoi polygons based on a simple geometric centroid equivalent, better suited for an integrated programmatic solution. All the data presented in this appendix was calculated based on the equivalent Voronoi polygons.

The following tables reflect the segmentation by gender identified by Carra [52]. They also include the rule identified in Munro et al. [183] where Mountain bears were defined as those for which 80% or more of their core home range* was above an elevation of 1,700 m, while the rest of the animals were categorized as Foothills bears.

All elevations are in meters (m), and all areas are given in square kilometers (km^2).

Table C.3: Bear population including minimum and maximum elevation and median area of their geometric area ranges.

	Elevation			
Bears	Min	Max	Mean	Median area
105	218	3145	1606	107.5613

Table C.4: Bear population by gender including minimum and maximum elevation and median area of their geometric area ranges.

		Elevation			
Bears	Gender	Min	Max	Mean	Median area
56	Female	661	3145	1685	70.2418
49	Male	218	3116	1487	198.6406

Table C.5: Bear population by elevation including minimum and maximum elevation and median area of their geometric area ranges.

		Elevation			
Bears	Bear type	Min	Max	Mean	Median area
78	Foothills	218	3034	1376	145.4345
27	Mountain	1087	3145	2096	66.8554

The publication by Munro et al. [183] concentrated on a grizzly bear population on west-central Alberta, where the minimum elevation was around 700 m. The current work encompasses a much larger study area (section 3.2) and from additional examination of the previous tables the classification of Mountain and Foothills bears might not have

*Calculated for this work with the R geometric area function for collections of points in 2D [36]

Table C.6: Bear population by gender and elevation including minimum and maximum elevation and median area of their geometric area ranges.

			Elevation			
Bears	Gender	Bear type	Min	Max	Mean	Median area
36	Female	Foothills	661	2800	1396	87.8906
20	Female	Mountain	1121	3145	2112	60.3864
42	Male	Foothills	218	3034	1355	233.1173
7	Male	Mountain	1087	3116	2045	97.2347

enough classes to properly describe the grizzly bear groups considered in this work. An alternative classification fell outside the objectives of this work and was not attempted.

C.4 Temporal distribution of GPS data.

The tables in this section (C.7 to C.22) are a summary of the temporal distribution of all GPS points from all bears included in the study. The dates in the tables aggregating data by week of the year are all approximated, and meant to reflect the two significant periods for the collaring effort. The capture period starts in mid-April and ends in mid-May. The denning period starts in mid-October of one year and ends in mid-March (emergence) of the following year.

Table C.7: Temporal distribution of GPS points aggregated by week of the year, 1999-2003 (dates are approximated).

		1999		2000		2001		2002		2003	
Week	Date	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
1										2	4
2											
3											
4											
5										1	1
6											
7											
8								1	1		
9											
10											
11	Mid-Mar									1	1
12										1	1
13										2	4
14								1	1	2	14
15	Mid-Apr			2	42			3	13	6	58
16				3	78	3	53	4	51	7	83
17				4	101	5	104	5	90	8	101
18		1	15	6	135	6	182	8	141	10	129
19		2	47	7	183	10	295	9	207	11	184
20	Mid-May	7	166	10	243	12	361	11	262	13	296
21		7	239	14	344	13	429	15	310	16	384
22		12	313	16	395	15	453	17	413	20	409
23		12	368	17	452	15	440	16	389	20	428
24		11	329	17	475	17	484	16	397	23	454
25		11	356	18	461	16	510	16	331	25	473
26		11	325	18	467	15	478	15	366	27	515
27		11	247	18	420	15	506	15	406	26	544

Table C.8: Temporal distribution of GPS points aggregated by week of the year, 1999-2003 (continued, dates are approximated).

Week	Date	1999		2000		2001		2002		2003	
		Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
28		10	275	17	361	15	472	17	359	26	455
29		10	293	14	369	16	444	17	367	27	455
30		10	313	15	357	16	429	16	391	26	451
31		10	291	12	334	15	431	16	360	25	472
32		10	272	12	327	16	476	13	325	23	446
33		9	262	12	277	16	521	13	279	23	423
34		9	266	13	290	15	539	12	285	22	403
35		8	221	13	309	16	437	12	252	21	392
36		8	189	13	304	14	435	11	219	21	465
37		7	203	14	288	14	412	11	264	21	463
38		6	177	13	325	14	402	11	245	21	449
39		6	173	11	303	13	451	14	259	20	441
40		5	157	9	222	14	437	12	250	20	455
41	Mid-Oct	4	146	8	231	15	439	11	207	19	460
42		4	133	7	225	15	381	13	282	19	367
43		4	115	8	234	12	304	15	324	15	324
44		3	99	7	186	9	163	13	282	15	299
45		3	51	5	117	8	107	12	184	12	152
46		3	35	5	63	8	117	9	126	10	87
47		1	29	5	31	9	50	7	111	3	64
48		1	11	4	40	3	43	6	104	2	33
49		1	5	1	15	1	3	6	41	2	9
50								3	33		
51						1	1	3	28		
52						1	2	1	23	1	1
53								1	11		

Table C.9: Temporal distribution of GPS points aggregated by week of the year, 2004-2008 (dates are approximated).

		2004		2005		2006		2007		2008	
Week	Date	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
1				1	6	1	1	1	1		
2				1	28	1	1			1	1
3											
4											
5											
6											
7											
8											
9											
10											
11	Mid-Mar										
12		3	6			1	1			1	1
13		5	33			1	1			2	6
14		5	51			2	90	1	12	3	66
15	Mid-Apr	7	48	6	18	4	316	2	66	4	111
16		8	82	6	71	6	491	2	186	6	208
17		9	111	7	74	7	704	2	206	5	70
18		11	162	8	88	7	723	2	210	6	547
19		14	183	13	193	7	690	2	205	6	751
20	Mid-May	13	200	14	555	8	759	6	413	8	903
21		15	214	13	520	9	906	9	737	10	1,110
22		15	267	15	576	11	1,277	8	865	13	1,055
23		17	270	18	1,614	12	1,705	8	789	12	1,141
24		21	259	19	2,297	12	1,996	10	877	14	1,239
25		23	368	21	3,100	15	1,959	11	976	14	1,180
26		25	491	22	3,014	14	1,879	10	917	14	1,169
27		23	505	21	3,017	15	2,067	10	814	14	999

Table C.10: Temporal distribution of GPS points aggregated by week of the year, 2004-2008 (continued, dates are approximated).

		2004		2005		2006		2007		2008	
Week	Date	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
28		24	571	22	3,119	14	2,347	9	811	12	1,054
29		24	614	22	3,444	14	2,168	9	764	12	1,037
30		24	600	20	3,217	11	1,905	8	702	12	964
31		22	560	18	2,861	10	1,857	8	671	11	1,032
32		22	546	16	2,979	10	1,961	9	745	11	973
33		20	546	16	3,011	10	1,716	8	596	10	1,023
34		21	507	14	3,046	8	1,573	7	640	11	962
35		19	440	14	3,145	8	1,646	7	635	12	978
36		19	482	13	2,964	8	1,626	8	690	10	1,000
37		19	487	13	3,080	8	1,246	8	677	12	846
38		18	447	12	3,090	6	1,329	7	720	10	839
39		19	459	12	2,597	6	1,399	7	775	8	874
40		16	468	11	2,187	6	767	7	759	9	715
41	Mid-Oct	16	443	10	770	6	709	7	751	5	594
42		15	392	9	626	4	404	7	517	6	568
43		12	243	8	569	4	466	5	437	5	340
44		10	94	8	538	4	246	5	438	3	289
45		3	46	5	302	3	79	6	356	1	105
46		2	39	5	172	2	3	5	261		
47		2	43	2	67	1	2	6	228		
48		1	42	2	11	1	2	4	115		
49		1	42					1	7		
50		1	40					1	1		
51		1	42	1	1	1	2	1	1		
52		1	42								
53		1	36								

Table C.11: Temporal distribution of GPS points aggregated by month, 1999-2003.

	1999		2000		2001		2002		2003	
Month	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
January									3	5
February							1	1		
March									2	10
April	1	10	6	384	5	200	7	202	9	314
May	12	880	17	1,387	15	1,537	17	1,226	21	1,340
June	13	1,432	19	1,970	17	2,052	16	1,602	26	2,024
July	10	1,255	18	1,574	16	2,040	19	1,685	28	2,073
August	10	1,106	14	1,332	16	2,158	14	1,280	24	1,863
September	7	774	14	1,215	14	1,824	14	1,054	21	1,951
October	4	542	9	941	16	1,596	17	1,204	19	1,677
November	3	122	5	201	10	375	13	599	12	369
December					2	9	6	136	3	6
Total points		6,121		9,004		11,791		8,989		11,632
Max bears	13		19		17		19		28	

Table C.12: Temporal distribution of GPS points aggregated by month, 2004-2008.

	2004		2005		2006		2007		2008	
Month	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points	Bears	GPS points
January			1	34	1	2	1	1	1	1
February										
March	6	67			2	2			3	14
April	11	397	8	251	7	1,701	2	527	6	716
May	17	971	19	2,239	10	3,548	8	2,087	13	4,098
June	28	1,607	25	11,780	16	7,989	11	3,809	15	5,008
July	24	2,553	22	13,935	15	9,258	10	3,368	13	4,500
August	23	2,244	18	13,462	10	7,781	9	2,903	13	4,382
September	19	1,998	13	12,080	8	6,058	8	3,053	13	3,768
October	17	1,319	10	2,737	6	2,499	7	2,611	7	2,135
November	3	181	5	448	3	179	6	1,067	2	128
December	1	184	1	1	1	2	2	3		
Total points		11,521		56,967		39,019		19,429		24,750
Max bears	28		25		16		11		15	

Table C.13: Temporal distribution of GPS points by individual bear, 1999.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G001				10	159	80						
G002					137	162	149	160	165	154	14	
G003					105	145	142	148	79			
G004					105	152	148	140	157	91		
G006					89	130	114	92				
G008					90	120	94	70	19			
G010					37	160	137	146	73			
G012					23	6						
G013					3	92	95	3				
G016					10	132	113	129	143	159	10	
G017					21	45						
G020						59	130	120	138	138	98	
G029					101	149	133	98				

Table C.14: Temporal distribution of GPS points by individual bear, 2000.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G001				53	146	152	133	102	43			
G002					135	105	27	1	78			
G004				14	124	118	65					
G006					83	115	7					
G007				78	46	31	5					
G010					14	137	127	131	119	90		
G011					28	124	16	23	15			
G012				134	117	79	52					
G014					37	87	72	69	18			
G016						51	86	83	63	42	7	
G017					28	49						
G020				93	153	142				53	12	
G023				12	123	97	102	106	102	81		
G024					96	134	122	133	101			
G026					20	59	49	50	68			
G027					80	149	153	153	145	163	43	
G028					125	168	149	143	148	129		
G029					32	107	144	82	73	4		
G033						66	220	172	192	274	136	
G034							45	84	50	105	3	

Table C.15: Temporal distribution of GPS points by individual bear, 2001.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G002					2	137	51	78	133	92		
G003					115	152	150	150	96	61		
G004					27	133	105	121	103	53		
G010					83	88						
G012					118	143	112	117	121	142	25	
G014				60	50							
G016						84	93	126	120	56		
G017					117	33						
G020				37	153	149	129	74		45	62	
G023						169	298	235		9	5	
G024					147	153	163	158	164	172	89	6
G028							61	148	146	102	2	3
G029				17	252	225	248	284	261	208	30	
G033				60	175	111	120	128	124	146	69	
G034				26								
G036					135	112	114	89	72	69	3	
G038					52	67	51	82	95	91		
G040					48	70	25	11	7	12		
G042						111	206	229	259	202	9	
G100					63	115	114	128	123	136	81	

Table C.16: Temporal distribution of GPS points by individual bear, 2002.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G002					2	8	2					
G003							31	119	88	126	99	106
G007										43	42	
G008					106	124	114	127	111	131	134	4
G010						84	152	121	117	46		
G011					41	81	32					
G012		1		78	124	101	98	36		73	95	10
G017					258	201	223	76		82	29	
G020				55	107	125	118	5				
G023					27	62	82	65	43	64	5	1
G027				25	40	52	50	23	28	29	1	
G028				27	92	177	155	144	114	145	17	
G029					13							
G033				9	130	113	111	119	96	97	100	14
G035					53	131	107	113	106	19		
G036				6	23							
G037					29	94	75	79	81	83	18	
G040				2	130	100	109	113	104	114	17	
G042							69	140	134	118	1	
G048										4		
G050					46	69	71		23			
G054					5	80	24		8	7		
G100							62		1	23	41	1

Table C.17: Temporal distribution of GPS points by individual bear, 2003.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G003	3				33	117	73	106	117	116	6	
G007				37	30		71					
G010				17	54	74	65	73	62	50		
G012			7	85	143	97	84	45				
G023	1			28	80	63	75	51	67	34		
G028	1			45	153	149	129	93	152	143		
G033			3	69	116	95	77	50				
G037							24	11				
G040				30	107	110	97	83	112	104	22	
G043				1	121	111	114	128	132	143	18	
G044					108	107	69	80	105	52		
G048				2	15	30	31	27	37	24		
G055					47	108	103	101	84			
G057					63	110	59					
G058					1	76	100	93	94	96	34	
G060					103							
G061					65	103	144	141	152	69		
G062						50	62	48	66	80	3	
G065						42	60	62	81	71	15	
G066					14	60	37					
G067					13	94	81	101	72	81	67	1
G068					2	93	40					
G070						63	100	96	115	132	39	
G071						42	75	87	89	71		
G072						44	90	102	102	122	13	
G073						19	51	48	61	75	2	
G074						7	33	57	46			
G100					62	71	57	83	87	114	41	1
G106					10	89	72	97	118	100	109	4

Table C.18: Temporal distribution of GPS points by individual bear, 2004.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G004							380	440	498	301		
G048				1	11	12						
G059				27	92	54	68	93	82	65		
G062			7	56	6							
G064					65	53	43	72	36	29		
G065			1	19	23	26	36	16	52	30		
G067			2	36	79	35						
G071				16	45	48	59	75	83	35		
G072			20	87	96	80	57	53	30			
G073				19	25	13	46	50	37	31		
G075F						61	136	126	163	133		
G076					81	76	63	8				
G077					88	91	93	117	94	41		
G078					38	70	78	49	34			
G080					46	122	112	55		43	172	184
G081						23	137	145	161	130		
G086					15	47	62	37	64	26		
G087						18	29					
G088						22	31	21	7			
G091						89	83	100	104	49		
G092						76	78	93	81	56	8	
G093						4						
G095						28	31	42	39	14		
G096						37	55	68	66	9		
G097						35	72	88	82	92	1	
G098						232	550	400				
G099						80	155	83	285	235		
G100			16	51	58	28						
G106			21	84	80	51						
G89K				1	123	96	99	13				

Table C.19: Temporal distribution of GPS points by individual bear, 2005.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G059				57	108	80	98	36				
G064				15	30							
G080	34											
G086				4	16	9	18					
G087				76	91	98	99	114	116	134	23	
G091				34	64	13						
G092				21	67	12						
G093					17	64	120	88				
G095				2	4	11	13					
G097				42	69	39	48	7				
G201					84	81	72	101	9			
G203					568	1971	2012	2044	1978	115		
G205						4	71	116	85	104	20	
G207					75	9	69	72	80			
G208					110	123	115	66				
G209					193							
G210					479	2064	2039	2037	1562	125		
G216					179	874						
G217					10	125	119	32				
G218					68	2015	2074	2098	2071	700	343	1
G219					7	131	121					
G225						38	22	35	102	62		
G226						141	143	167	172	138		
G227						91	48					
G228						1423	2022	2064	1999	592	7	
G229						1158	1963	1982	1992	673	55	
G230						177	639	349	559	94		
G231						1029	2010	2054	1355			

Table C.20: Temporal distribution of GPS points by individual bear, 2006.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G008				170	561	498	576	672	608	621	71	
G040				294	649	478	498					
G200						176	455	581	196			
G204						116	429	495	329	464	79	
G205			1	39	80	233	581	271				
G211						132	473	557	311	398	29	2
G213						57	41					
G218	2		1	545	684	2034	2095	2111	2067	265		
G225				8	39	25	32					
G228				130	403	455	407	248				
G229				515	640	1886	1956	2007	1950	257		
G230						676	557					
G235					276	62						
G236					56	362	495	498	495	494		
G237						318	352	341	102			
G241					160	481	311					

Table C.21: Temporal distribution of GPS points by individual bear, 2007.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G008				233	333	290	151	41				
G110									218	308	130	
G211	1			294	544	481	55					
G223					337	582	356	426	557	505	144	1
G224					251	416	435	158	399	195		
G226					112	444						
G238					254	400	430	350	289	397	349	2
G260					173	386	399	457	468	527	36	
G262					83	229	229	126				
G264						147	355	361	136			
G265						279	519	449	547	317	104	
G266						155	439	535	439	362	304	

Table C.22: Temporal distribution of GPS points by individual bear, 2008.

Bear	January	February	March	April	May	June	July	August	September	October	November	December
G077					676	660	680	699	618	81		
G085								24	605			
G090								165	143	222		
G110			3	148	333	342	165	234	266	661	123	
G111						225	298	197	156			
G223			6	170	672	629	598	506	315	356		
G224				33	386	423	347	400	247	66		
G230					13	287	239	302	199			
G238			5	205	514	340	19					
G251						120	404	16	150	414	5	
G253					204	436	415	575	511			
G254					110	184	252	305	63			
G260	1			86	619	501	413	391	322	335		
G262					28	368	507	568	173			
G264					11	328						
G265				74	459	135						
G266					73	30	163					

Appendix D

Most frequent patterns, frequency tables for nine sub variations.

This appendix contains the tables of most frequent patterns for all tree variations except the original asymmetric and the completed asymmetric, which were included in subsection 4.2.4 for a more detailed analysis.

D.1 Original symmetric.

Table D.1: Original symmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
222	1,042	555555	7,016	555555555	4,185	555555555555	2,887
444	345	333333	2,695	333333333	1,388	333333333333	779
000	165	777777	2,132	777777777	1,070	777777777777	593
141	86	222222	440	555555551	292	555555555551	135
666	59	333335	270	155555555	270	222222222222	125
161	53	533333	231	222222222	227	155555555555	113
244	52	777771	226	333333335	135	555555555511	86
442	47	333331	224	533333333	106	115555555555	84
144	46	177777	223	555555557	94	333333333335	77
441	45	111777	216	444444444	91	533333333333	47
422	36	133333	205	777777771	86	555555555557	45
224	34	111333	200	177777777	84	111111111121	43
414	26	333111	199	111117777	75	777777777771	39
142	25	211111	189	333333353	75	177777777777	37
1A1	23	111112	184	755555555	75	333333333355	37
241	16	117777	183	133333333	73	755555555555	35

Table D.2: Original symmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
555555555555555	2,161	55555555555555555555	1,456
333333333333333	435	33333333333333333333	167
777777777777777	340	77777777777777777777	143
222222222222222	72	22222222222222222222	33
555555555555551	72	15555555555555555555	25
155555555555555	57	55555555555555555511	23
333333333333335	49	55555555555555555553	21
555555555555553	44	35555555555555555555	17
355555555555555	41	11155555555555555555	14
115555555555555	37	53333333333333333333	11
555555555555511	36	55555555555555155555	10

Table D.3: Original symmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
5555555555555555555555555	1,099	555555555555555555555555555555	873
3333333333333333333333333	86	777777777777777777777777777777	52
7777777777777777777777777	74	333333333333333333333333333333	47
5555555555555555555555511	13		
5555555555555555555555553	11		

Table D.4: Original symmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
5555555555555555555555555555555555555555	559
7777777777777777777777777777777777777777	33

D.2 Distinct asymmetric.

Table D.5: Distinct asymmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
141	110	717171	247	171717171	76	717171717171	29
161	59	171717	240	717171717	70	171717171717	28
424	54	131313	185	131313131	46	313131313131	9
242	52	313131	185	313131313	34	131313131313	7
414	44	171715	74	121212121	19	242424242424	7
142	30	517171	74	151717171	17	515151717171	7
1A1	27	121212	73	171717151	17	717171717173	4
214	26	212121	72	424242424	15	121212121212	3
241	20	171713	44	212121212	14	131313131373	3
412	18	717131	39	717171713	13	171717171737	3
143	16	131317	38	717171737	13	271717171717	3
5A5	13	717173	37	131313151	10	121212151513	2
616	13	317171	36	171717371	10	151717171515	2
316	12	171313	34	717171715	10	151717171713	2
613	11	717175	33	151313131	9	171571717171	2
617	11	713131	32	171717173	9	171717371717	2

Table D.6: Distinct asymmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
171717171717171	15	71717171717171717171	5
424242424242424	4	17171717171717171717	4
131313131313131	2	12121241317137317131	1
171271717171717	2	12131315151313131313	1
717171717371713	2	12141213131313131757	1
121212121251717	1	12371251512121515151	1
121212121313157	1	12542121321212121212	1
121212173121752	1	13131312121274121721	1
121213132143232	1	13131313142142424242	1
121213145412171	1	13131373137373131213	1
121215713573731	1	13137315715127171717	1

Table D.7: Distinct asymmetric, most frequent patterns, n = {25, 30}

$n = 25$	
Pattern	Support
1241317131217172521712151	1
1313131313131312171217131	1
1317171317131371737173751	1
1371515751745414121213212	1
1417217151717171721512171	1

$n = 30$	
Pattern	Support
1241317131217172521712151	1
1313131313131312171217131	1
1317171317131371737173751	1

D.3 Distinct symmetric.

Table D.8: Distinct symmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
141	110	717171	247	171717171	76	717171717171	29
161	59	171717	240	717171717	70	171717171717	28
424	54	131313	185	131313131	46	313131313131	9
242	52	313131	185	313131313	34	131313131313	7
414	44	171715	74	121212121	19	242424242424	7
142	30	517171	74	151717171	17	515151717171	7
1A1	27	121212	73	171717151	17	717171717173	4
214	26	212121	72	424242424	15	121212121212	3
241	20	171713	44	212121212	14	131313131373	3
412	18	717131	39	717171713	13	171717171737	3
143	16	131317	38	717171737	13	271717171717	3
5A5	13	317171	36	131313151	10	121212151513	2
616	13	171313	34	171717371	10	151717171515	2
316	12	717175	33	717171715	10	151717171713	2
613	11	713131	32	151313131	9	171571717171	2
617	11	131717	30	171717173	9	171717371717	2

Table D.9: Distinct symmetric, most frequent patterns, n = {15, 20}

$n = 15$	
Pattern	Support
171717171717171	15
424242424242424	4
131313131313131	2
171271717171717	2
717171717371713	2
121212121251717	1
121212121313157	1
121212173121752	1
121213132143232	1
121213145412171	1
121215713573731	1

$n = 20$	
Pattern	Support
171717171717171	15
424242424242424	4
131313131313131	2
171271717171717	2
717171717371713	2
121212121251717	1
121212121313157	1
121212173121752	1
121213132143232	1
121213145412171	1
121215713573731	1

Table D.10: Distinct symmetric, most frequent patterns, n = {25, 30}

$n = 25$	
Pattern	Support
1241317131217172521712151	1
1313131313131312171217131	1
1317171317131371737173751	1
1371515751745414121213212	1
1417217151717171721512171	1

$n = 30$	
Pattern	Support
121242321321212123132312135212	1
131312121274121721215312151572	1
171371241317131217172521712151	1

D.4 Artificial, repeated asymmetric.

Table D.11: Artificial, repeated asymmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
222	1,042	555555	7,020	555555555	4,185	555555555555	2,887
444	345	333333	2,696	333333333	1,389	333333333333	780
221	301	777777	2,138	777777777	1,074	777777777777	594
122	269	222222	440	555555551	292	555555555551	135
141	86	333335	271	155555555	270	222222222222	125
666	71	533333	231	222222222	227	155555555555	113
161	53	777771	227	333333335	136	555555555511	86
244	52	177777	224	533333333	106	115555555555	84
442	47	333331	224	555555557	94	333333333335	78
144	46	111777	216	444444444	91	555555555515	73
441	45	133333	205	777777771	87	555555155555	63
422	36	111333	200	177777777	85	555555515555	62
224	34	333111	200	111117777	75	555555551555	62
424	34	117777	183	333333353	75	555555555155	61
414	26	333311	183	755555555	75	555515555555	59
142	25	113333	174	133333333	73	555551555555	57

Table D.12: Artificial, repeated asymmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
555555555555555	2,161	55555555555555555555	1,456
333333333333333	435	33333333333333333333	167
777777777777777	340	77777777777777777777	143
222222222222222	72	22222222222222222222	33
555555555555551	72	55555555555555555551	30
155555555555555	57	15555555555555555555	25
333333333333335	50	55555555555555555511	23
555555555555553	44	55555555555555555553	21
355555555555555	41	55555555555555555111	20
115555555555555	37	11555555555555555555	18
555555555555511	36	33333333333333333335	17

Table D.13: Artificial, repeated asymmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
5555555555555555555555555	1,099	555555555555555555555555555555	873
3333333333333333333333333	86	777777777777777777777777777777	52
7777777777777777777777777	74	333333333333333333333333333333	47
2222222222222222222222222	19		
5555555555555555555555111	12		

Table D.14: Artificial, repeated asymmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
555555555555555555555555555555	873
777777777777777777777777777777	52

D.5 Artificial, repeated symmetric.

Table D.15: Artificial, repeated symmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
222	1,042	555555	7,020	555555555	4,185	555555555555	2,887
444	345	333333	2,696	333333333	1,389	333333333333	780
221	301	777777	2,138	777777777	1,074	777777777777	594
122	269	222222	440	555555551	292	555555555551	135
141	86	333335	271	155555555	270	222222222222	125
666	71	533333	231	222222222	227	155555555555	113
161	53	777771	227	333333335	136	555555555511	86
244	52	177777	224	533333333	106	115555555555	84
442	47	333331	224	555555557	94	333333333335	78
144	46	111777	216	444444444	91	444444444444	51
441	45	133333	205	777777771	87	533333333333	47
422	36	111333	200	177777777	85	555555555557	45
224	34	333111	200	111117777	75	111111111121	43
424	34	211111	189	333333353	75	777777777771	39
414	26	111112	184	755555555	75	177777777777	38
142	25	117777	183	133333333	73	333333333355	38

Table D.16: Artificial, repeated symmetric, most frequent patterns, n = {15, 20}

$n = 15$	
Pattern	Support
555555555555555	2,161
333333333333333	435
777777777777777	340
222222222222222	72
555555555555551	72
155555555555555	57
333333333333335	50
555555555555553	44
355555555555555	41
115555555555555	37
555555555555511	36

$n = 20$	
Pattern	Support
55555555555555555555	1,456
33333333333333333333	167
77777777777777777777	143
22222222222222222222	33
55555555555555555551	30
15555555555555555555	25
55555555555555555511	23
55555555555555555553	21
55555555555555555111	20
11555555555555555555	18
33333333333333333335	17

Table D.17: Artificial, repeated symmetric, most frequent patterns, n = {25, 30}

$n = 25$	
Pattern	Support
5555555555555555555555555	1,099
3333333333333333333333333	86
7777777777777777777777777	74
2222222222222222222222222	19
5555555555555555555555111	12

$n = 30$	
Pattern	Support
555555555555555555555555555555	873
777777777777777777777777777777	52
333333333333333333333333333333	47

Table D.18: Artificial, repeated symmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
5555555555555555555555555555555555555555	559
7777777777777777777777777777777777777777	33

D.6 Artificial, distinct asymmetric.

Table D.19: Artificial, distinct asymmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
121	597	717171	247	171717171	76	171717171717	28
212	324	171717	240	717171717	70	313131313131	9
141	110	131313	185	131313131	46	424242424242	8
161	59	313131	185	313131313	34	242424242424	7
424	54	171715	74	121212121	19	515151717171	7
242	52	517171	74	151717171	17	717171717173	4
414	44	121212	73	171717151	17	121212121212	3
142	30	212121	72	424242424	15	131313131373	3
615	27	171751	45	212121212	14	171717371713	3
214	26	171713	44	717171713	13	271717171717	3
124	24	157171	43	717171737	13	513131313131	3
241	20	131317	39	131313151	10	151517171715	2
412	18	717131	39	515171717	10	171571717171	2
421	17	317171	37	717171715	10	171717371717	2
143	16	717173	37	151313131	9	212121213131	2
616	13	171313	34	171717173	9	212121515131	2

Table D.20: Artificial, distinct asymmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
171717171717171	15	71717171717171717171	5
424242424242424	4	12152521212121252151	1
131313131313131	2	12371251512121515151	1
151513131313131	2	12423213212121231323	1
171271717171717	2	12542121321212121212	1
271717171717171	2	13131212127412172121	1
717171717371713	2	13131313131313121712	1
121212121251717	1	13131313142142424242	1
121212173121752	1	13171717171717537371	1
121213131357175	1	13232142121317151571	1
121213145412171	1	13571315151212121323	1

Table D.21: Artificial, distinct asymmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
1241317131217172521712151	1	121242321321212123132312135212	1
1313131313131312171217131	1	131312121274121721215312151572	1
1317171317131371737173751	1	171371241317131217172521712151	1
1371515751745414121213212	1		
1417217151717171721512171	1		

D.7 Artificial, distinct symmetric.

Table D.22: Artificial, distinct symmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
121	597	717171	247	171717171	76	171717171717	28
212	324	171717	240	717171717	70	313131313131	9
141	110	131313	185	131313131	46	424242424242	8
161	59	313131	185	313131313	34	242424242424	7
424	54	171715	74	121212121	19	515151717171	7
242	52	517171	74	151717171	17	717171717173	4
414	44	121212	73	171717151	17	121212121212	3
142	30	212121	72	424242424	15	131313131373	3
214	26	131315	52	212121212	14	171717371713	3
124	24	171751	45	717171713	13	271717171717	3
241	20	171713	44	717171737	13	513131313131	3
412	18	157171	43	131313151	10	131515131313	2
421	17	131317	39	515171717	10	151517171715	2
143	16	717131	39	717171715	10	171571717171	2
616	13	513131	38	151313131	9	171717371717	2
316	12	317171	37	171717173	9	212121213131	2

Table D.23: Artificial, distinct symmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
171717171717171	15	71717171717171717171	5
424242424242424	4	12152521212121252151	1
131313131313131	2	12371251512121515151	1
151513131313131	2	12423213212121231323	1
171271717171717	2	12542121321212121212	1
271717171717171	2	13131212127412172121	1
717171717371713	2	13131313131313121712	1
121212121251717	1	13131313142142424242	1
121212173121752	1	13171717171717537371	1
121213131357175	1	13232142121317151571	1
121213145412171	1	13571315151212121323	1

Table D.24: Artificial, distinct symmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
1241317131217172521712151	1	121242321321212123132312135212	1
1313131313131312171217131	1	131312121274121721215312151572	1
1371515751745414121213212	1	171371241317131217172521712151	1
1417217151717171721512171	1		
1515157175171717571717171	1		

D.8 Completed asymmetric.

Table D.25: Completed asymmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
000	1,175,046	000000	893,234	000000000	676,131	000000000000	484,317
500	29,110	100000	43,407	000100000	36,064	000000000100	30,133
005	28,467	000001	41,864	000010000	34,780	000000100000	29,519
333	3,443	000500	16,809	000001000	34,637	000100000000	29,224
777	3,425	500000	16,503	000000100	33,477	100000000000	29,193
200	1,985	000050	16,404	100000000	32,764	010000000000	29,090
002	1,935	000005	16,396	000000010	32,066	000000000010	29,026
222	655	050000	16,254	010000000	31,979	001000000000	28,981
400	417	005000	16,181	000000001	31,968	000000000001	28,568
004	398	000300	8,015	001000000	31,907	000000010000	28,297
444	295	300000	7,825	000500000	14,028	000000001000	28,157
A00	290	000030	7,787	000050000	13,695	000010000000	27,921
00A	286	000003	7,778	000005000	13,647	000001000000	27,895
600	243	030000	7,745	000000500	13,445	000000000500	12,123
006	234	003000	7,718	000000050	12,927	000000500000	11,969
211	225	000700	7,370	000000005	12,880	000000000050	11,865

Table D.26: Completed asymmetric, most frequent patterns, n = {15, 20}

$n = 15$		$n = 20$	
Pattern	Support	Pattern	Support
000000000000000	387,070	00000000000000000000	268,907
000000000100000	27,403	00000000010000000000	23,388
000100000000000	26,966	00000000000100000000	23,089
000000000010000	26,401	00000000100000000000	23,085
000000100000000	26,311	00000000001000000000	22,320
000010000000000	26,211	00000000000000100000	12,133
000001000000000	26,141	00000000000010000000	11,718
000000000001000	25,937	00000100000000000000	11,221
000000010000000	25,042	00000000000001000000	10,935
000000001000000	24,909	00000010000000000000	10,723
000000000000100	14,546	00000001000000000000	10,575

Table D.27: Completed asymmetric, most frequent patterns, n = {25, 30}

$n = 25$		$n = 30$	
Pattern	Support	Pattern	Support
0000000000000000000000000	180,878	000000000000000000000000000000	127,241
0000000000000010000000000	10,271	000000000000000000100000000000	7,580
0000000000000100000000000	10,124	000000000000000000001000000000	7,169
0000000000100000000000000	9,503		
0000000000010000000000000	9,410		

Table D.28: Completed asymmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
0000000000000000000000000000000000000000	58,295
0000000000000000000000000000010000000000	2,759

D.9 Completed symmetric.

Table D.29: Completed symmetric, most frequent patterns, n = {3, 6, 9, 12}

$n = 3$		$n = 6$		$n = 9$		$n = 12$	
Pattern	Support	Pattern	Support	Pattern	Support	Pattern	Support
000	1,175,046	000000	893,234	000000000	676,131	000000000	676,131
500	29,110	100000	43,407	000100000	36,064	000100000	36,064
005	28,467	000001	41,864	000010000	34,780	000010000	34,780
333	3,443	000500	16,809	000001000	34,637	000001000	34,637
777	3,425	500000	16,503	000000100	33,477	000000100	33,477
200	1,985	000050	16,404	100000000	32,764	100000000	32,764
002	1,935	000005	16,396	000000010	32,066	000000010	32,066
222	655	050000	16,254	010000000	31,979	010000000	31,979
400	417	005000	16,181	000000001	31,968	000000001	31,968
004	398	000300	8,015	001000000	31,907	001000000	31,907
444	295	300000	7,825	000500000	14,028	000500000	14,028
A00	290	000030	7,787	000050000	13,695	000050000	13,695
00A	286	000003	7,778	000005000	13,647	000005000	13,647
600	243	030000	7,745	000000500	13,445	000000500	13,445
006	234	003000	7,718	000000050	12,927	000000050	12,927
211	225	000700	7,370	000000005	12,880	000000005	12,880

Table D.30: Completed symmetric, most frequent patterns, n = {15, 20}

$n = 15$	
Pattern	Support
000000000000000	387,070
000000000100000	27,403
000100000000000	26,966
000000000010000	26,401
000000100000000	26,311
000010000000000	26,211
000001000000000	26,141
000000000001000	25,937
000000010000000	25,042
000000001000000	24,909
000000000000100	14,546

$n = 20$	
Pattern	Support
00000000000000000000	268,907
00000000010000000000	23,388
00000000000100000000	23,089
00000000100000000000	23,085
00000000001000000000	22,320
00000000000000100000	12,133
00000000000010000000	11,718
00000100000000000000	11,221
00000000000001000000	10,935
00000010000000000000	10,723
00000001000000000000	10,575

Table D.31: Completed symmetric, most frequent patterns, n = {25, 30}

$n = 25$	
Pattern	Support
0000000000000000000000000	180,878
0000000000000010000000000	10,271
0000000000000100000000000	10,124
0000000001000000000000000	9,503
0000000000100000000000000	9,410

$n = 30$	
Pattern	Support
000000000000000000000000000000	127,241
000000000000000000100000000000	7,580
000000000000000000000100000000	7,169

Table D.32: Completed symmetric, most frequent patterns, n = 40

$n = 40$	
Pattern	Support
0000000000000000000000000000000000000000	58,295
0000000000000000000000000000010000000000	2,759