Agglomerative 2-3 Hierarchical Classification: Theoretical and Applicative Study

Sergiu Theodor Chelcea

UNIVERSITÉ DE NICE SOPHIA ANTIPOLIS – UFR Sciences

Ecole Doctorale des Sciences et Technologies de l'Information et de la Communication

THÈSE

pour obtenir le titre de

Docteur en Sciences
Spécialité Informatique

présentée et soutenue par

Sergiu Theodor Chelcea

Agglomerative 2-3 Hierarchical Classification: Theoretical and Applicative Study

Thèse dirigée par Jacques LEMAIRE et Brigitte TROUSSE

et préparée à l'INRIA Sophia Antipolis, projet AxIS

Rapporteurs: Mlle. Maria Paula BRITO
M. Djamel A. ZIGHED

soutenue le 23 Mars 2007

Jury :

M.	Patrice	BERTRAND	Maître de Conférences	Examinateur
Mlle.	Maria Paula	BRITO	Professeur Associé	Rapporteur
M.	Jacques	LEMAIRE	Professeur	Co-Directeur de thèse
M.	Michel	RUEHER	Professeur	Président
Mlle.	Brigitte	TROUSSE	Chargée de recherche	Co-Directeur de thèse
M.	Djamel A.	ZIGHED	Professeur	Rapporteur
M.	Yves	LECHEVALLIER	Directeur de recherche	Invité

UNIVERSITÉ DE NICE SOPHIA ANTIPOLIS – UFR Sciences

Ecole Doctorale des Sciences et Technologies de l'Information et de la Communication

THÈSE

pour obtenir le titre de

Docteur en Sciences
Spécialité Informatique

présentée et soutenue par

Sergiu Theodor Chelcea

Classification Ascendante 2-3 Hiérarchique : étude théorique et applicative

Thèse dirigée par Jacques LEMAIRE et Brigitte TROUSSE

et préparée à l'INRIA Sophia Antipolis, projet AxIS

Rapporteurs: Mlle. Maria Paula BRITO
M. Djamel A. ZIGHED

soutenue le 23 Mars 2007

Jury :

M.	Patrice	BERTRAND	Maître de Conférences	Examinateur
Mlle.	Maria Paula	BRITO	Professeur Associé	Rapporteur
M.	Jacques	LEMAIRE	Professeur	Co-Directeur de thèse
M.	Michel	RUEHER	Professeur	Président
Mlle.	Brigitte	TROUSSE	Chargée de recherche	Co-Directeur de thèse
M.	Djamel A.	ZIGHED	Professeur	Rapporteur
M.	Yves	LECHEVALLIER	Directeur de recherche	Invité

Abstract

Classification is one of the many fields in Data Mining which aims at extracting information from large data volumes by using different computational techniques from machine learning, statistics and pattern recognition. One of the two common approaches in the unsupervised classification (or clustering) is the hierarchical clustering. Its purpose is to produce a tree in which the nodes represent clusters of the initial analyzed data. One of the main drawbacks of the most known and used hierarchical agglomerative method, the Agglomerative Hierarchical Classification (AHC), is the fact that it cannot highlight groups of objects with characteristics from two or more classes, property found for example in overlapping clusters.

This thesis deals with a recent extension of the Agglomerative Hierarchical Classification, the Agglomerative 2-3 Hierarchical Classification (2-3 AHC), proposed by P. Bertrand in 2002, with a focus on its application to the Data Mining fields. The three major contributions of this thesis are: the theoretical study of the 2-3 hierarchies (also called paired hierarchies), the new 2-3 AHC algorithm and its implementation, and the first applicative study of this method in two Data Mining fields.

Our theoretical study includes the discovery of four new theoretical properties of the 2-3 hierarchies and the definition of the aggregation links between clusters for this type of structure. This allowed us to highlight a special case of clusters merging and to introduce an intermediate step in the 2-3 hierarchies' construction. The systematic and exhaustive study of possible cases leads us to formulate the best choices in term of linkage and structure indexing, in order to improve the quality of the 2-3 hierarchies.

Next, based on our theoretical study and contributions, we proposed a new general Agglomerative 2-3 Hierarchical Classification algorithm. This represents the result of our previous study: a powerful algorithm exploring the multiple possibilities of the 2-3 hierarchical model. A theoretical complexity analysis of our 2-3 AHC algorithm, showed a reduced complexity from $\mathcal{O}(n^3)$ in the initial algorithm, to $\mathcal{O}(n^2 \log n)$ for our algorithm. The tests on different datasets (real and generated) confirmed our theoreti-

cal complexity study. Very satisfying results were obtained by analyzing the "quality" of the 2-3 hierarchies compared with the traditional hierarchies: up to 50% additional created clusters and a maximal gain of 84% using the Stress index.

We also proposed an object-oriented model of our algorithm that was integrated in the "Hierarchical Clustering Toolbox" (HCT), a toolbox that we developed for the visualization of the agglomerative hierarchical classification methods. We also integrated this model as a method of case indexing in the Case Based Reasoning platform, CBR*Tools, developed at INRIA Sophia Antipolis, and used it to design recommender systems.

Our last contribution lies in the first study of the applicability of the 2-3 AHC on real data from two Data Mining fields: Web Mining and XML Document Clustering. This study leads to interesting results and was based on the comparison of the 2-3 hierarchical clustering of INRIA's research teams using either the users' behavior on their Web sites, or their XML annual reports, with the existing structure of the research themes organization.

Finally, to conclude, we show that this subject is far from being exhausted and we propose several research perspectives related to the Agglomerative 2-3 Hierarchical Classification and to our HCT toolbox, developed during this thesis.

Keywords: *2-3 hierarchy, paired hierarchy, Agglomerative 2-3 Hierarchical Classification, Clustering, linkage, index, HCT toolbox, Web Mining, XML Document Clustering*

Résume

La classification est l'un des nombreux domaines de la Fouille de Données qui vise à extraire l'information à partir de grands volumes de données en utilisant différentes techniques computationnelles de l'apprentissage, des statistiques et de la reconnaissance des formes. Une des deux approches fondamentales de la classification non supervisée (ou clustering) est la classification hiérarchique. Son but est de produire un arbre dans lequel les nœuds représentent des classes des objets analysés. L'un des inconvénients principaux de la méthode ascendante hiérarchique la plus connue et la plus utilisée, la Classification Ascendante Hiérarchique (CAH), est le fait qu'on ne peut pas mettre en évidence de classes d'objets ayant des caractéristiques communes. Cette propriété se trouve par exemple dans les classes qui se recouvrent et qui ont été introduites et étudiées dans les extensions de la CAH.

Cette thèse porte sur une extension récente de la Classification Ascendante Hiérarchique, appelée Classification Ascendante 2-3 Hiérarchique et proposée par P. Bertrand en 2002, avec en vue son application au domaine de la Fouille de Données. Les trois contributions majeures de cette thèse résident dans l'étude théorique des 2-3 hiérarchies (appelées aussi *paired hierarchies*), dans le nouvel algorithme de 2-3 CAH avec son implémentation, et dans la première étude applicative de cette méthode dans deux domaines de la Fouille de Données.

Notre étude théorique inclut la découverte de quatre nouvelles propriétés théoriques des 2-3 hiérarchies et les définitions des liens d'agrégation entre les classes pour ce type de structure. Ceci nous a aussi permis de mettre en évidence un cas spécial de fusion des classes et d'introduire une étape intermédiaire dans la construction des 2-3 hiérarchies. L'étude exhaustive et systématique des cas possibles nous a permis de formuler les meilleurs choix concernant le lien d'agrégation et l'indexation de la structure, avec en vue l'amélioration de la qualité des 2-3 hiérarchies.

Dans un deuxième temps, basé sur notre étude et contributions théoriques, nous proposons un nouvel algorithme général de Classification Ascendante 2-3 Hiérarchique.

Ceci représente la concrétisation de notre travail précédent, aboutissant à un algorithme performant, qui explore plusieurs possibilités du modèle 2-3 hiérarchique. Une analyse théorique de la complexité de notre algorithme a montré que la complexité a été réduite de $\mathcal{O}(n^3)$ dans l'algorithme initial de 2-3 CAH à $\mathcal{O}(n^2 \log n)$ pour notre algorithme. Les comparaisons des 2-3 hiérarchies avec les hiérarchies classiques obtenues sur différents ensembles de données (réels et simulés), ont validé l'analyse de complexité par les temps d'exécution. En plus, des résultats très satisfaisants ont été obtenus en analysant la "qualité" des 2-3 hiérarchies comparées aux hiérarchies classiques : jusqu'à 50% de classes en plus, et un gain maximum de 84% en utilisant l'indice de Stress. Nous avons ensuite proposé un modèle orienté-objet de notre algorithme de 2-3 CAH, qui a été intégré dans une boite à outils "Hierarchical Clustering Toolbox" (HCT) que nous avons développée pour la visualisation des méthodes ascendantes hiérarchiques de classification. Ce modèle a été également intégré comme méthode d'indexation des cas dans la plateforme de Raisonnement à Partir de Cas (RàPC), CBR*Tools, développé à l'INRIA Sophia Antipolis, et utilisé pour la conception des systèmes de recommandations.

Notre dernière contribution concerne une toute première étude de l'utilisation de notre algorithme de 2-3 CAH sur des données réelles relevant de deux domaines de la Fouille des Données : le Web Mining et la Classification de Documents XML. Celle-ci a donné lieu à des résultats intéressants et portait sur la comparaison de la classification 2-3 hiérarchique des équipes de recherche de l'INRIA en utilisant soit le comportement des utilisateurs sur leur sites Web, soit leur rapport annuel d'activité écrit en XML, par rapport à la structure organisationnelle existante en thèmes de recherche.

Pour conclure, nous montrons que ce sujet est loin d'être épuisé et nous proposons plusieurs pistes de recherche future relatives à la Classification Ascendante 2-3 Hiérarchique ainsi qu'à notre boite à outils HCT, développée pendant cette thèse.

Mots Clefs: *2-3 hiérarchie, paired hierarchy, Classification Ascendante 2-3 Hiérarchique, Classification, lien, indice, boite à outils HCT, Web Mining, Classification de Documents XML*

I dedicate this thesis to my parents, Cristiana and Ion Chelcea.

Acknowledgements

I would like to thank, first and foremost, my advisors, Professor Jacques Lemaire and Brigitte Trousse, for providing me with the unique opportunity to work in the research area of classification in the AxIS team. My sincere gratitude and appreciation goes to Brigitte Trousse, for her guidance and mentorship, and for her encouragement and support during the completion of this thesis. She helped me a lot in clarifying many uncertain points, in suggesting interesting and fruitful research directions and in creating a comfortable work environment for me and the other members of our team.

I am especially grateful to Patrice Bertrand for supervising a major part of the theoretical study of this thesis, for the long discussions on the 2-3 hierarchies, his availability and fruitful advices. This work would have been less complete without his precious collaboration.

Special thanks go to Maria Paula de Pinho de Brito Duarte Silva and Djamel Abdelkader Zighed for their careful reading of this thesis and thoughtful comments and for agreeing to examinate my thesis despite their busy schedule.

I would like to express my sincere gratitude to Michel Rueher for accepting to be the President of the jury for my thesis defense.

I would like to thank Yves Lechevallier for his input and assistance on the tests and for his interest in this research.

I would like to thank Doru for the interesting discussions we had in the coffee corner, Mihai for his help on the Hierarchical Clustering Toolbox development, Semi for his "man and google" help and all other colleagues in the AxIS research team, including Florent, Hicham, Alice, Calin, Sophie and Bernard for their friendship and companionship.

On a personal note, the love and encouragement of my entire family and Florina

kept me going through it all. I dedicate this thesis with all my heart to my parents, Mica and Ta'anelu. I also would like to thank Ilie for his valuable support in the difficult moments.

I would finally like to thank all of my friends for their company and support, and for making me laugh when I needed to.

Table of Contents

1 Introduction **3**

1.1 Classical Agglomerative Hierarchical Classification (AHC) 5

1.2 Extensions of the Classical AHC 7

1.2.1 Pyramidal Classification 7

1.2.2 Weak Hierarchies 9

1.2.3 Agglomerative 2-3 Hierarchical Classification (2-3 AHC) 10

1.2.4 Summary 11

1.3 Motivations and Thesis Objectives 12

1.4 Thesis Document Structure 13

I Contributions to Hierarchical Clustering **17**

2 Theoretical Study of the 2-3 AHC **19**

2.1 Classical AHC: Method and Algorithm 20

2.1.1 Definitions and Properties 24

2.1.2 Examples 25

2.2 Agglomerative 2-3 Hierarchical Classification (2-3 AHC) 26

2.2.1 Definitions and Properties 26

2.2.2 Examples 29

2.2.3 2-3 AHC Initial Algorithm 31

2.3 Four New 2-3 AHC Properties 33

2.3.1 The case of two clusters that properly intersect 33

2.3.2 Candidate Clusters Elimination 35

2.3.3 Intermediate Merging 36

2.3.4 Integrating the Refinement Step into the Merging Step 39

2.4 Aggregation Index 42

2.4.1 Context 43

2.4.2 Link 44

2.5 2-3 Hierarchy Indexing 47

2.6 Proper Intersection Analysis 49
2.6.1 Inversion problem 49
2.6.2 *Blind Merging* 50
2.7 2-3 AHC Algorithm Execution 51
2.7.1 Proper Intersection During Algorithm Execution 51
2.7.2 Blind Merging's Influence 53
2.8 Complete Link Definitions 54
2.9 Discussion and Perspectives 55

3 A New 2-3AHC Algorithm 57
3.1 Initial 2-3 AHC Algorithm 58
3.2 Using the New Theoretical Properties 60
3.2.1 Adding an Intermediate Merge at the End of each β Merging . . 60
3.2.2 2-3 AHC Algorithm's New Formulation 61
3.2.3 Reformulating the Update of the Set $\mathcal{M}_i$ of Candidates 63
3.2.4 Facts 65
3.2.5 Integration of the Refinement Step into the Merging Step 65
3.3 Proposition of a New 2-3 AHC Algorithm 66
3.4 Complexity Analysis of our 2-3 AHC Algorithm 69
3.4.1 Data Matrix Indexing 69
3.4.2 Theoretical Complexity Calculus 72
3.4.3 Experimental Validation 73
3.5 Experimental Qualitative Comparison between AHC and 2-3 AHC . . . 75
3.5.1 Ruspini Dataset 76
3.5.2 Urban Itineraries 79
3.5.3 Artificially Generated Data 85
3.5.4 Abalone Dataset 87
3.6 Comparison between 2-3 AHC and APC 90
3.7 Discussion and Perspectives 93

4 Toolbox for Hierarchical Clustering Methods and CBR*Tools Integration 95
4.1 Design of an Object-Oriented Model 95
4.1.1 Notion of an Object-Oriented Framework 96
4.1.2 Object-Oriented Model 97
4.2 Toolbox for Hierarchical Clustering Methods 99
4.2.1 Data Selection and Representation 99
4.2.2 Methods Selection and Execution 101
4.2.3 Results Visualization and Analysis 102
4.3 Integration of the 2-3 AHC in the CBR*Tools Framework 105

4.3.1 CBR*Tools: a Framework for Case Based Reasoning 105
4.3.2 HAC23Index Integration for Memory Organization 106
4.4 Applications 109
4.5 Discussion and Perspectives 112

II Study of applying the 2-3 AHC in the Web Mining and Document Clustering Fields 113

5 Web Mining Application 119
5.1 Introduction 120
5.1.1 WUM Terms 120
5.1.2 Log Files 121
5.1.3 WUM steps 122
5.2 Data Description and Motivations 123
5.3 Extended Relational DB Model for Data Preprocessing 125
5.3.1 Data Preprocessing 125
5.3.2 Extension of the Relational DB Model 127
5.4 Research Teams Clustering Based on Web Users Behaviours 133
5.4.1 Advanced Data Preprocessing 133
5.4.2 Analyses and Results 136
5.5 Discussion and Perspectives 143

6 XML Document Clustering Application 145
6.1 Introduction 145
6.2 INRIA's Activity Reports Analysis 147
6.2.1 Data Description and Motivations 147
6.2.2 Data Preprocessing 148
6.2.3 Dissimilarity Matrix Generation 150
6.2.4 Analyses and Results 151
6.3 Other Applications 156
6.4 Discussion and Perspectives 156

7 Conclusions and Perspectives 161
7.1 Main Contributions of this Thesis 161
7.1.1 Theoretical Study of the 2-3 AHC 162
7.1.2 A New 2-3 AHC Algorithm 162
7.1.3 2-3 AHC Applications 164
7.2 Future Works and Perspectives 165

Bibliography 167

A Example of the *Blind Merging*'s Influence **179**

B Single Link and "Normal" Execution of 2-3 AHC Algorithm **183**

C Tests on Simulated Data **187**
C.1 Execution Times and Complexity 187
C.2 Stress Gain 190

D Use of HAC23Index as Part of the CBR*Tools in an CBR Application **197**
D.1 Reasoning System Construction 198
D.2 Case Representation 198
D.3 Memory organization 199
D.4 Object oriented implementation 200

E INRIA Research Teams Organization **203**

F Be-TRIP Recommender System **207**

List of Figures

1.1 Example of a classical hierarchy 6
1.2 Pyramid 8
1.3 Classical Hierarchy 8
1.4 Non-closed weak hierarchy 9
1.5 Quasi-hierarchy 9
1.6 Example of a 2-3 hierarchy 11
1.7 Classical Hierarchies Extensions 11

2.1 Dendrogram 23
2.2 Horizontal dendrogram of a hierarchy representation 24
2.3 Dissimilarity matrix example 25
2.4 Hierarchy partitioning 25
2.5 Example of a 2-3 hierarchy 26
2.6 Example of a *proper intersection*: $Xp \cap Y = Z$ 27
2.7 Example of levels inversion 29
2.8 Using the *double single-link* 29
2.9 Example of an 2-3 Hierarchy 30
2.10 AHC and 2-3 AHC 30
2.11 Dataset 30
2.12 Single-link levels inversion 31
2.13 The *double single link* 31
2.14 Complete-link levels inversion 31
2.15 *Extended indexing formula* 31
2.16 The γ condition 32
2.17 α and β merging when X and Y *properly intersect* each other 34
2.18 f value for β case 35
2.19 f value for α case 35
2.20 Successor candidates elimination 36
2.21 Successor case of a *proper intersection* 37
2.22 Predecessor case of a *proper intersection* 38

2.23 Evolution in the common successor case . . . 39
2.24 Refinement influence . . . 40
2.25 $f(X) < f(Y) < f(Z)$ . . . 41
2.26 $f(X) = f(Y) \leq f(Z)$ . . . 41
2.27 $f(X) = f(Y) = f(Z)$ . . . 41
2.28 $f(X) < f(Y) = f(Z)$ . . . 42
2.29 Clusters on the same level in a 2-3 hierarchy . . . 44
2.30 No intersection . . . 46
2.31 Proper intersection . . . 46
2.32 Levels inversion . . . 46
2.33 Average-link and 2-3 AHC algorithm . . . 47
2.34 Using the *extended indexing formula* . . . 48
2.35 Using the *double single-link* . . . 48
2.36 Proper intersection in a 2-3 hierarchy . . . 50

3.1 Intermediate merging . . . 60
3.2 Intermediate merging . . . 61
3.3 α merging of X_i and Y_i, with $X_i \cap Y_i = \emptyset$ . . . 63
3.4 α merging of X_i and Y_i, with $X_i \cap Y_i \neq \emptyset$ . . . 64
3.5 β merging of X_i and Y_i . . . 64
3.6 Refinement example (Fact 3.2.5) . . . 65
3.7 Data structure example . . . 70
3.8 2D Points . . . 71
3.9 Classical AHC . . . 71
3.10 2-3 AHC V3 avoiding BM diss . . . 71
3.11 2-3 AHC V3 avoiding BM all . . . 71
3.12 SL average execution times . . . 74
3.13 SL average complexity . . . 74
3.14 CL average execution times . . . 74
3.15 CL average complexity . . . 74
3.16 Ruspini dataset . . . 78
3.17 The 2-3 hierarchy on the selected points from Figure 3.16 . . . 78
3.18 The classical hierarchy on the selected points from Figure 3.16 . . . 78
3.19 Classical AHC on the 40 urban itineraries . . . 81
3.20 2-3 AHC with intermediate merging (2-3 AHC V2) on the 40 itineraries 82
3.21 2-3 AHC avoiding blind merging (2-3 AHC V3) on the 40 urban itineraries 83
3.22 Initial 2-3 AHC (2-3 AHC ini) on the 40 urban itineraries . . . 84
3.23 CL avg. created clusters number . . . 86
3.24 SL avg. created clusters number . . . 86
3.25 SL avg. Stress gain with confidence intervals . . . 87

3.26 SL max. Stress gain . . . 87
3.27 CL avg. Stress gain . . . 88
3.28 CL min. Stress gain . . . 88
3.29 Single link's average Stress gain on the Abalone dataset samples . . . 88
3.30 CL avg. Stress gain on the Abalone dataset . . . 89
3.31 CL min. Stress gain on the Abalone dataset . . . 89
3.32 CL max. Stress gain on the Abalone dataset . . . 89
3.33 Pyramid on 40 urban itineraries . . . 91
3.34 2-3 AHC with intermediate merging (2-3 AHC V2) on 40 urban itineraries 92

4.1 Object oriented model of the agglomerative hierarchical algorithms . . . 98
4.2 HCT: Data matrix generation from DB . . . 100
4.3 HCT: DB server selection . . . 100
4.4 Hierarchical Clustering Toolbox: Input Data Representation . . . 101
4.5 HCT: Choosing the linkage and the algorithm to execute . . . 102
4.6 HCT: Output Structure Graphical Representation . . . 103
4.7 HCT: Selection of a clustering partition . . . 104
4.8 HCT: Induced dissimilarites and initial data matrices analysis . . . 104
4.9 Case Based Reasoning . . . 106
4.10 HAC23Index hot-spots and their integration in CBR*Tools . . . 108
4.11 Urban Itinerary Creation and Visualization . . . 111

5.1 A Web Request from INRIA's Web Server Log (ECLF Format) . . . 122
5.2 Schema of a General WUM Process . . . 123
5.3 The data preprocessing steps . . . 125
5.4 DB Relational Model for Log Files from [Tan05a] . . . 128
5.5 Extended DB Relational Model for Log Files . . . 129
5.6 URL Generalization . . . 130
5.7 2-3 Hierarchy on theme 3 projects during Per_1 . . . 139
5.8 2-3 hierarchy on theme 3 projects during Per_2 . . . 140
5.9 2-3 hierarchy on theme Cog projects during Per_2 . . . 141
5.10 Classical hierarchy on theme 3 projects in Per_1 . . . 142
5.11 2-3 hierarchy on theme 3 projects in Per_1 . . . 143

6.1 AxIS team in K-all analysis . . . 151
6.2 AxIS team in T-PF analysis . . . 151
6.3 Atypical teams in T-P experiment . . . 152
6.4 Atypical teams in K-all experiment . . . 152
6.5 Atypical teams in T-C experiment . . . 152
6.6 AHC in T-P experiment . . . 155
6.7 2-3 AHC in T-P experiment . . . 155

A.1 The data set . . . 179
A.2 AHC dissimilarities . . . 179
A.3 The classical hierarchy . . . 179
A.4 2-3 AHC dissimilarities . . . 180
A.5 The created 2-3 hierarchy . . . 180
A.6 The resulting ultrametric . . . 180
A.7 The resulting 2-3 ultrametric . . . 180

B.1 The data set . . . 183
B.2 Initial steps . . . 183
B.3 Double indexing formula and the normal execution *(i)* . . . 183
B.4 Levels inversion for normal execution *(i)* of the 2-3 AHC algorithm . . . 184
B.5 2-3 AHC algorithm with integrated refinement *(ii)* . . . 184

C.1 SL maximum execution times . . . 188
C.2 SL maximum complexity . . . 188
C.3 CL maximum execution times . . . 188
C.4 CL maximum complexity . . . 188

D.1 CBR reasoning cycle in the CAR sample application . . . 198

List of Tables

2.1 Coefficients values for different links 22
2.2 Dissimilarity matrix example for α and β case 35

3.1 Example of lexicographical criteria influence 71
3.2 The single-link results on the Ruspini dataset 77
3.3 The complete-link results on the Ruspini dataset 77
3.4 Complete link results on the professional itineraries classification 85

5.1 Most Visited Topics on INRIA's Web Sites on Per_1 130
5.2 Data preprocessing 133
5.3 Summary of data preprocessing 134
5.4 Binary table for Per_1 describing the navigations using the visited topics 135
5.5 Quantities used for topics similarity computation 136
5.6 INRIA's Web site topics clustering using 2-3 AHC for Per_1 138

6.1 Size of the data for the five experiments 150
6.2 Documents (teams) representation using words frequency 150
6.3 2-3 AHC V3 clustering in T-C experiment using Jaccard and CL 154
6.4 Complete-link Stress gains of 2-3 AHC using the simple distance 155

E.1 National Web pages presenting the research themes organization 204
E.2 Teams from theme 3 in the old and new research themes 205

List of Abbreviations and Terminology

Some of the abbreviations and acronyms used throughout this thesis are listed below:

2-3 AHC	Agglomerative 2-3 Hierarchical Classification
AHC	Agglomerative Hierarchical Classification
APC	Ascendent Pyramidal Classification
SL	Single Link
CL	Complete Link
CBR	Case-Based Reasoning
DM	Data Mining
HCT	Hierarchical Clustering Toolbox
BM	Blind Merging
GIS	Geographical Information System
DB	Database
DS	Data Structure
HTTP	Hypertext Transfer Protocol
INRIA	The French National Institute for Research in Computer Science and Control
IP	IP Address
KDD	Knowledge Discovery in Databases
LCS	Longest Common Substring
UA	User Agent
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WUM	Web Usage Mining
WWW	World Wide Web

Introduction

Chapter 1

Introduction

"*... le seul moyen de faire une méthode instructive et naturelle,*
c'est de mettre ensemble les choses qui se ressemblent,
et de séparer celles qui diffèrent les unes des autres."

Georges Louis Leclerc, Comte de Buffon, *Histoire Naturelle, 1749*

("*... the only way to have an instructive and natural method,*
is to put together things that resemble,
and to separate from each other the things that are differents.")

Sometimes it is difficult to imagine how revolutionary an idea was, especially when that idea is today accepted as common knowledge. One of these, is the above well-known phrase of the famous naturalist Georges Louis Leclerc (1707-1788), which is known today as one of the first known definitions of the *classification* concept.

Today, classification is one of the many fields in *Data Mining* (DM), also known as *Knowledge Discovery in Databases* (KDD) [FPSSU96], which aims at extracting information from large data volumes. In order to achieve this, data mining uses different computational techniques from machine learning, statistics and pattern recognition.

Unsupervised classification or *clustering* [Gor99, JMF99, Ber02a] is one of the most important fields in Data Mining and, as said before, consists in grouping together similar items into same groups while the dissimilar ones are asserted to different groups, without any apriori knowledge on the obtained groups. Alternatively, when the prior knowledge on the data is present, the technique is called *supervised classification* or *discriminant analysis*. These groups of objects are usually called *clusters* or *classes* and are used to synthesize the information contained in the initial data. In this case one can say that simplification is preferred to the details in the data.

Clustering has found lately a wide spread of applications in a large variety of fields, from archaeology and natural sciences, to bioinformatics, psychology and economics to

name only a few. Therefore, clustering merges and combines different techniques from different disciplines: statistics, computer science, mathematics, artificial intelligence, databases, etc. For instance, the continuous growth of the World Wide Web (WWW), has lead to the emergence of new applications domains, such as the *Web Mining*, that is: the application of DM techniques on Web data. This new field and some of its classification techniques are discussed later in Part II of this thesis.

The two common approaches in statistical clustering are *partitioning clustering* and *hierarchical clustering* [YKM99].

In the partitioning clustering one starts with the whole initial analyzed dataset, which is then split into k clusters. Usually the number k has to be specified before the analysis and is influencing the result of the clustering. There are also techniques to determine the most appropriate values of k, based usually on the selection of the "optimal" result (partition) for a range of values of k [MC85, Dub87].

The partitioning methods usually produce clusters by optimizing a criterion function, and sometimes due to the combinatorial number of possibilities, the algorithm is run repeatedly. Example of used criterion in this case include: the *squared error criterion*, the *diameter*, the *sum of distances*, etc.

Among the most known partitioning methods, we remind the *k-means* [Mac67], the *dynamic clustering* [Did73], the *minimum spanning trees* [GH85], etc. Interested reader can refer to [Gor99] for more details.

The hierarchical clustering methods can also be divided into *agglomerative*, *divisive* and *incremental* methods. The purpose of these types of classification is to produce a tree in which the nodes represent clusters of the initial analyzed set. In particular, the initial set is the root of the tree while the leaves represent the singletons (clusters with one element). This type of structure gives thus an enhanced visual representation than the partitioning methods. The investigator can then select the suitable partitioning from its point of view, by making a trade off between the number of clusters and their homogeneity degree.

In the *incremental hierarchical methods*, an already constructed classification of objects is augmented by successively inserting new objects into the classification [Sib73, BHT05]. The interest of these kind of methods lie in their capacity to analyze very large datasets. BIRCH [ZRL96], is a well-known incremental hierarchical classification algorithm having an overall complexity of $\mathcal{O}(n)$.

In the *hierarchical divisive methods*, starting from a single cluster (of all objects), successive splits of clusters are performed to obtain smaller clusters [Rao71, GHJ91]. The two main problems of this methodology are: *which cluster to split?*, and *how to split it?*. Casual strategies in this case are the followings: split the highest cardinality cluster, split each cluster at a given level or split the one with the largest intra-cluster variance.

In the *agglomerative hierarchical methods*, starting from the initial elements (the *singletons*), the clusters are successively merged into higher level clusters, until the entire set of analyzed objects becomes a cluster. These resulting hierarchical structures contain a number of partitions which can be then easily visualized using a graphic.

By far, the most known and used hierarchical agglomerative method is the *Agglomerative Hierarchical Classification*. Sometimes in different studies this method is also called the *Ascending Hierarchical Classification*. In the rest of this document we will denote it as the classical Agglomerative Hierarchical Classification or simply the classical AHC method. This method has been extensively studied [SS63, Ben73, JD98] during the last decades and has a wide number of applications in many different fields, as bioinformatics, signal processing, web mining, etc.

In AHC, the iterative merging of clusters is made by minimizing a *cost function* between clusters. This cost function is usually known as the *aggregation index* or the *linkage*. But sometimes multiple pairs of clusters can minimize this cost function, and choosing between them can have a big influence on the result [Har83]. The main characteristic of this technique is that a succession of nested partitions is obtained. This means that all clusters are either disjoint or included one in another, during the final result interpretation.

But sometimes data can contain objects with characteristics from two or more classes, property found for example in overlapping clusters. For instance, a work can often belong to two different genres in document classification. Obtaining such clusters using the classical AHC is not possible, and thus extensions of the classical AHC have been lately proposed: the *pyramids* [Did73], the *weak-hierarchies* [BD89], the *k-weak hierarchies* [Dia97] or more recently the 2-3 *hierarchies* [Ber02d] to name a few. Overlapping clusters present in these types of structures, can provide far more information for the investigator's analysis compared to the disjoint clusters in the classical hierarchies, but sometimes these structures can become very difficult to interpret especially on larger datasets.

We resume next the classical AHC technique and its AHC algorithm in Section 1.1, while the aforementioned extensions of the AHC are briefly presented and analyzed in the following Section 1.2.

1.1 Classical Agglomerative Hierarchical Classification (AHC)

Generally speaking, the main goal of the classical Agglomerative Hierarchical Classification, is to create a hierarchy of nested partitions over an initial set of elements. The set of elements can be a set of individuals or a set of variables, which are two-by-two comparable.

Definition 1 : *A collection $\mathcal{C}$ of non-empty parts of a finite set E is called a* **hierarchy** *if:*

1. $E \in \mathcal{C}$;

2. $\forall x \in E$ we have that $\{x\} \in \mathcal{C}$ (terminal nodes or singletons);

3. $\forall (X, Y) \in \mathcal{C} \times \mathcal{C}$ we have that $X \cap Y \in \{\emptyset, X, Y\}$ (we say that X and Y are hierarchical).

□

Sokal and Sneath proposed in 1963 [SS63] the first version of the classical AHC algorithm, consisting in two phases: initialization and merging.

During the first phase, the singletons dissimilarity matrix is computed using a chosen dissimilarity measure, while the singletons represent the initial set of clusters. We remind that a dissimilarity is a non-negative mapping δ defined on the couples of analyzed elements satisfying the reflexivity ($\forall\, x : \delta(x, x) = 0$) and symmetry ($\forall\, x, y : \delta(x, y) = \delta(y, x)$) conditions. A distance is a dissimilarity satisfying the triangular condition: $\forall\, x, y, z : \delta(x, y) \leq \delta(x, y) + \delta(y, z)$.

In the second phase of the algorithm, successive mergings are performed between the two *closest clusters*, until the initial objects are all merged into a final cluster. The two clusters are closest in the sense of a chosen *aggregation link*, (i.e. *single link*, *complete link*, *average link*, etc.).

At the end of the second phase, the resulting structure is a sequence of nested partitions which can be visualized using a graphic called *dendrogram*.

A small example of a classical hierarchy is presented in Figure 1.1.

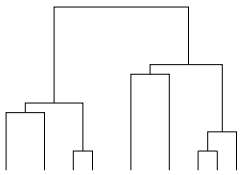


Figure 1.1: Example of a classical hierarchy

A hierarchy will induce a new dissimilarity matrix over the initial elements based on the dissimilarity at which they were first regrouped in a cluster in the hierarchy. This induced dissimilarity matrix is an *ultrametric*. Formally, an ultrametric is a distance satisfying the *ultrametric inequality*: $d(x, z) < max\{d(x, y), d(y, z)\}$. This inequality is also referred as the *strong triangle* inequality.

The induced matrix can be then compared with the initial matrix or with the induced matrices of other hierarchical methods, for quality analysis [SS73a].

The AHC algorithm has an $\mathcal{O}(n^2 \log n)$ complexity and usually uses as input a dissimilarity matrix on the initial elements. More details on the classical AHC and its

algorithm are presented in Section 2.1.

1.2 Extensions of the Classical AHC

As we saw, the classical hierarchies consist in sequence of nested clusters, which are strictly included or disjoint. But usually the clusters contained within the analyzed data can also have common parts (properties) that are not emphasized within the classical hierarchies. For this, different classification methods providing overlapping clusters have been proposed: the *additive clustering* [SA79] and the B_k method [JS68] as non-hierarchical methods, while the *pyramids* [Did84], the *weak hierarchies* [BD89], the *k-weak hierarchies* [Dia97] and the *2-3 hierarchies* [Ber02d] are all extensions of the classical AHC.

In the following we will make a short analysis of the methods that extend the hierarchical framework.

1.2.1 Pyramidal Classification

The Ascendent Pyramidal Classification (denoted APC in the rest of the text) was proposed in 1984 by Diday [Did84] in order to generalize the classical AHC. The result in this case is a *pyramid*, a structure containing "overlapping" clusters (non-disjoint or non-hierarchical clusters). In a pyramid a cluster can overlap at most two other clusters. A pyramid is also denoted as a *pseudo-hierarchy* [Fic84]. This type of structure is richer than a classical hierarchy, and represents better the initial analyzed data.

Definition 2 : *Let be E a finite set and $\mathcal{C}$ a set of non-empty parts of E (called nodes or clusters), $\mathcal{C}$ is a* **pyramid** *if it has the following properties:*

1. $E \in \mathcal{C}$;
2. $\forall x \in E$ *we have that* $\{x\} \in \mathcal{C}$ *(terminal nodes or singletons);*
3. $\forall (X, Y) \in \mathcal{C} \times \mathcal{C}$ *we have that* $X \cap Y \in \mathcal{C}$ *or* $X \cap Y = \emptyset$;
4. *A total order θ exists in E compatible with $\mathcal{C}$.* □

The pyramidal model was introduced in order to achieve the notion of compatibility between a dissimilarity d and an order θ. We say that d and θ are compatible iif $\forall$ ordered triplet $x_i\,\theta\,x_j\,\theta\,x_k$, we have: $d(x_i, x_k) \geq max\{d(x_i, x_j), d(x_j, x_k)\}$. The clusters in a pyramid are intervals of the order θ, and can have common elements (individuals). In a pyramid the number of linear orders on the initial set E can be 2 or more, compared to the 2^{n-1} compatible orders for a classical hierarchy.

Knowing that clusters in a pyramid can not have more than two predecessors [Did84], the maximal number of non-singleton clusters is $\frac{n(n-1)}{2}$ compared to $n-1$ in a classical hierarchy ($n = |E|$).

As in the case of the hierarchies, an indexing value f can be associated to each cluster in a pyramid, to obtain an *indexed pyramid*, for which:

- $f(x) = 0$ where $x \in E$;
- $\forall X, Y \in \mathcal{C}, X \subset Y \Rightarrow f(X) \leq f(Y)$.

A *weakly indexed pyramid* is a indexed pyramid in which if $X \subset Y$ and $f(X) = f(Y)$ implies that X has two predecessors. Also, a pyramid is *strictly indexed* if $X \subset Y$ implies that $f(X) < f(Y)$.

The hierarchies are a particular case of pyramids, that is: when no cluster "overlaps" another cluster.

Different APC algorithms have been proposed lately, as a result of the different studies on the pyramidal classification [Ber86, Did86, Bri91, GS94, BJ97, Mfo98, GCA98, DBM01, RD05].

An example of a pyramid is presented in Figure 1.2. Compared with the classical hierarchy on the same dataset from Figure 1.3 the pyramid contains more clusters and gives a more accurate representation of the initial data. But for large datasets, the pyramids can become difficult to interpret [Did86], due to the big number of created clusters or to the investigators unfamiliarity with the pyramids.

As in the classical hierarchical case, by selecting an indexing level, a pyramid can be sectioned in order to obtain a clusters partition. The obtained clusters can be overlapped, highlighting thus common characteristics of different groups of individuals. This is no longer a partitioning, and we say that we obtain a covering or overlapping of the initial elements.

To ease the interpretation, different techniques of refining a pyramid based on the clusters indexing values have been proposed [RD04, RD05].

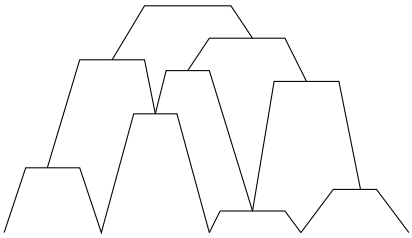


Figure 1.2: Pyramid

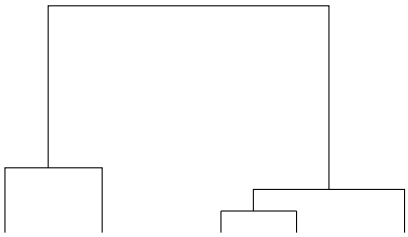


Figure 1.3: Classical Hierarchy

The APC algorithm [Did84, Ber86, Bri91, Mfo98] is more complex that the classical AHC one since at each step an "order" on the candidates clusters must be maintained, which implies additional tests and conditions. Its complexity is in $\mathcal{O}(n^3)$. The APC algorithm has been integrated into the European software, SODAS [EUR], which aims at extending the Statistics, the Data Mining and the Analysis of classic or complex data to concepts (symbolic data [Did87, BD00]).

The pyramidal classification has been applied in different fields, such as the bioinformatics [ADLCR99, Aud99] (genome analysis), or more generally in symbolic data analysis [BPZKC00].

1.2.2 Weak Hierarchies

The *weak hierarchies* were introduced in 1989 by Bandelt and Dress [BD89] and they generalized the classical hierarchies and also the pyramids. The main characteristics of a weak hierarchy is the fact that the intersection of any three clusters is the intersection of two of them. The same notion was also introduced in 1988 by Babedat [Bat88], under the name of *maximum medinclus*.

Definition 3 : *A* **weak hierarchy** *is a set $\mathcal{C}$ of non-empty parts of a finite set E such that:*

1. $E \in \mathcal{C}$;
2. $\forall x \in E$ *we have that* $\{x\} \in \mathcal{C}$ *(terminal nodes or singletons);*
3. $\forall X, Y, Z \in \mathcal{C} \Rightarrow X \cap Y \cap Z \in \{X \cap Y, X \cap Z, Y \cap Z\}$. □

A weak hierarchy $\mathcal{C}$ is said to be closed [Ban92, DF94] if it is closed under non-empty intersections: $\forall X, Y \in \mathcal{C}$ and $X \cap Y \neq \emptyset \Rightarrow X \cap Y \in \mathcal{C}$. A closed weak hierarchy is also called an *quasi-hierarchy*.

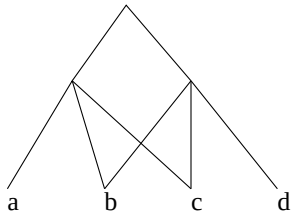


Figure 1.4: Non-closed weak hierarchy

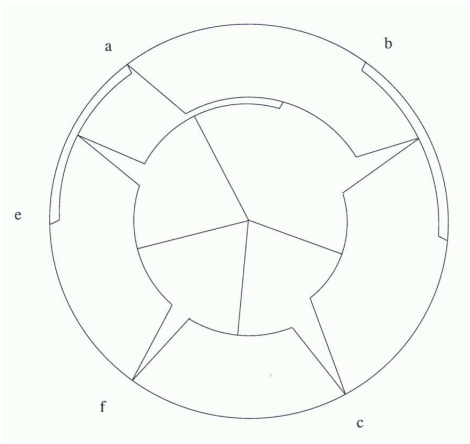


Figure 1.5: Quasi-hierarchy

An example of a non-closed weak hierarchy is given in Figure 1.4: the intersection of $\{abc\}$ with $\{bcd\}$ is $\{bc\}$ which is not a cluster of the weak hierarchy. A quasi-hierarchy is presented in Figure 1.5.

As in the case of the pyramids, the weak hierarchies contain more internal clusters (non-singletons), maximum $\mathcal{O}(n^2)$ [BD89, Bru01], than a classical hierarchy, giving

thus a better data representation. But in this case the visual representation of a weak hierarchy can be difficult to interpret due to the large number of clusters and due to the edges intersection (see Figure 1.4).

The mathematical properties of the weak hierarchies have been widely studied in the literature [BD94, Dia96, BJ97, Bru01, BBO04] and concern mainly the mathematical relations between the different types of weak hierarchies (closed, k-weak, etc) and the induced dissimilarity matrices (i.e. the quasi-ultrametrics, the weak k-ultrametrics). As in the case of pyramids, the induced dissimilarities are obtained from the different types of indexed weak hierarchies.

A generalization of the weak hierarchies to k intersections, has been proposed in [Dia97] and studied in [BJ03]. This kind of structure is called a *k-weak hierarchy*.

Definition 4 : *A* **k-weak hierarchy** *is a set $\mathcal{C}$ of non-empty parts of a finite set E such that:*

1. $E \in \mathcal{C}$;
2. $\forall x \in E$ *we have that* $\{x\} \in \mathcal{C}$ *(terminal nodes or singletons);*
3. $\forall X_1, X_2, ..., X_{k+1} \in \mathcal{C} \Rightarrow \cap_{i \in \{1,...,k+1\}} X_i \in \{\cap_{i \in \{\{1,...,k+1\}-j\}} X_i | 1 \leq j \leq k+1\}$.

□

The algorithm to construct such weak hierarchies has a complexity of $\mathcal{O}(n^4)$ [Bru05].

1.2.3 Agglomerative 2-3 Hierarchical Classification (2-3 AHC)

The Agglomerative 2-3 Hierarchical Classification (2-3 AHC) was recently proposed [Ber02d] by P. Bertrand*, and as in the pyramids and weak hierarchies case, it generalizes the classical Agglomerative Hierarchical Classification. The resulting structure is called a *2-3 hierarchy*† and it also allows overlapping clusters.

The main difference is that in a 2-3 hierarchy, a cluster can overlap only one other cluster. This restricts the number of internal clusters in a 2-3 hierarchy compared with the pyramids, but still gives a richer structure than a classical hierarchy.

Definition 5 : *A* **2-3 hierarchy** *is a set $\mathcal{C}$ of non-empty parts of a finite set E and closed under non-empty intersections such that:*

1. $E \in \mathcal{C}$;
2. $\forall x \in E$ *we have that* $\{x\} \in \mathcal{C}$ *(singletons);*
3. $\forall X \in \mathcal{C}$ *we have* $|\{Y \in \mathcal{C} : X \cap Y \notin \{X, Y, \emptyset\}\}| \leq 1$. □

*Paris-IX Dauphine University & AxIS INRIA Rocquencourt

†A *2-3 hierarchy* will be called in the future a *paired hierarchy* [Ber], but in the rest of this document we will use the initial terminology of 2-3 *hierarchy* introduced in [Ber02d]

An example of a 2-3 hierarchy is given in Figure 1.6.

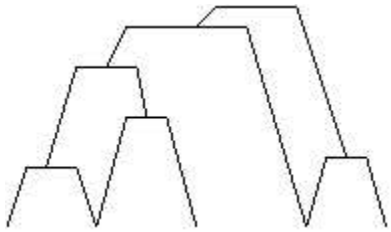


Figure 1.6: Example of a 2-3 hierarchy

The maximal number of internal clusters in a 2-3 hierarchy is $\lfloor \frac{3}{2}(n-1) \rfloor$, which represents 50% more than a classical hierarchy. This kind of structure is clearly generalized by the pyramids and the weak hierarchies, but its smaller number of internal clusters allows a smoother interpretation of the 2-3 AHC results.

The 2-3 AHC algorithm proposed in [Ber02d] has a $\mathcal{O}(n^3)$ complexity. Due to its recent nature, no additional study of these structures was made and no implementation of the 2-3 AHC algorithm was realized.

1.2.4 Summary

Figure 1.7 illustrates the extensions of the classical hierarchies presented so far and their inclusion. As we can see, the hierarchies are a particular case of 2-3 hierarchies, which in turn are a particular case of pyramids and so on.

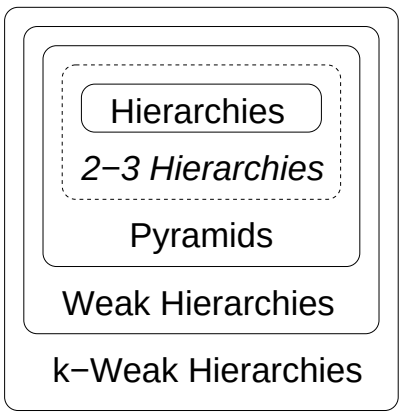


Figure 1.7: Classical Hierarchies Extensions

1.3 Motivations and Thesis Objectives

As we saw, extensions of the classical AHC technique have been proposed in the last decades to improve the data representation by allowing cluster overlapping. Such extensions include: the *pyramids*, the *k-weak hierarchies*, the 2-3 *hierarchies* etc. But their main drawbacks are the high complexity and the difficulty to interpret the results on larger datasets. Indeed, the number of created clusters is sometimes in $\mathcal{O}(n^2)$ (pyramids, k-weak hierarchies).

In the case of the most recent extension, the *Agglomerative* 2-3 *Hierarchical Classification* [Ber02d], the number of created clusters is relatively smaller, $\lfloor \frac{3}{2}(n-1) \rfloor$, while its complexity is $\mathcal{O}(n^3)$ compared to the $\mathcal{O}(n^2 \log n)$ of the classical AHC. This could make the 2-3 *hierarchies* easier to interpret, although the structure refinement might also be required when analyzing large datasets.

As we said in Section 1.2, the newly proposed 2-3 hierarchical structures are the only one with no additional theoretical study realized. Therefore some questions might arise:

- are there any other **theoretical properties of the 2-3 hierarchies**?
- **can the complexity of the 2-3 AHC algorithm be reduced**?

Moreover, the 2-3 AHC algorithm proposed in [Ber02d] **was not implemented** so far, which means that there are **no applications of the 2-3 AHC** whatsoever.

In this context, the objectives of this thesis are as follows:

- a theoretical study of the Agglomerative 2-3 Hierarchical Classification:
  - new properties of the 2-3 hierarchical structure;
  - a new 2-3 AHC algorithm based on the properties of the 2-3 hierarchies;
  - a complexity reduction of the 2-3 AHC algorithm;
  - a first implementation of a 2-3 AHC algorithm.
- an empirical study of the 2-3 AHC including:
  - the complexity validation;
  - comparison with the classical AHC.
- a first study of the applicability of the 2-3 AHC in Data Mining fields and in a recommender system context. From the DM fields, we chosen the recent research field of Web Mining, while the chosen recommender system is the *Be-TRIP* mobility recommender system that we proposed.

1.4 Thesis Document Structure

In this section we present the thesis document structure with a brief description of the different chapters according to our theoretical and applicative study:

Part I: Contributions to Hierarchical Clustering

In the first part of the thesis we will present our contribution related to the Agglomerative 2-3 Hierarchical Classification from the methodological and algorithmic point of view:

- **Chapter 2: Theoretical Study of the Agglomerative 2-3 Hierarchical Classification**

  In Chapter 2 we begin by presenting some used notions followed by the Agglomerative 2-3 Hierarchical Classification introduced in [Ber02d]. Next, the theoretical study of the 2-3 AHC will reveal three properties used later to propose a new 2-3 AHC algorithm. Also a study of the clusters dissimilarity measure (the *aggregation link*) is performed to define the single link and the complete link for the properly intersecting clusters. A special case of merging denoted the *blind merging* is described and studied.

- **Chapter 3: A New Agglomerative 2-3 Hierarchical Classification Algorithm**

  In Chapter 3 we propose a new Agglomerative 2-3 Hierarchical Classification general algorithm based on our previous theoretical study and contributions from Chapter 2. The main characteristic of this algorithm is an added *intermediate merging* step. Other optional steps include the *integrated refinement* and the *blind merging avoidance*. These optional steps give us four 2-3 AHC algorithm variants, which can create different 2-3 hierarchies on same the dataset.

  A theoretical complexity analysis of our 2-3 AHC algorithm proved that the complexity was reduced from $\mathcal{O}(n^3)$ in the initial 2-3 AHC algorithm to $\mathcal{O}(n^2 \log n)$ for our algorithm.

  Next, we tested the obtained 2-3 hierarchies and the classical hierarchy on different datasets (Ruspini [Rus69], urban itineraries [Bus05], simulated data, Abalone [Sam95]) for complexity execution times and structure quality. The obtained execution times verified our theoretical complexity of $\mathcal{O}(n^2 \log n)$. To determine the created structures quality, we have chosen the *Stress* coefficient [JW82] for comparing the initial data and the induced dissimilarity matrices. Using the complete link, we obtained an average gain of 23% (for the Stress) while the maximum gain was around 84% on the Abalone dataset.

- **Chapter 4: Toolbox for Hierarchical Clustering Methods and CBR*Tools Integration**

  In Chapter 4 of this thesis we present our object-oriented model of our 2-3 AHC algorithm which was implemented in Java. To better visualize, compare and analyze the hierarchies and the 2-3 hierarchies created on a same dataset, we developed the `Hierarchical Clustering Toolbox`. A brief presentation of the toolbox is then provided.

  Next, we integrated our 2-3 AHC algorithm in the Case-Based Reasoning (CBR) software library, CBR*Tools [Jac98], developed in our team. The purpose of this integration was the use of our method as a case indexing method in a CBR application, or more precisely in a recommender system to support information retrieval in a mobility context.

  As a part of the MobiVIP* project, we proposed a specification of such a system, the *Be-TRIP* mobility recommender system [CGT04, TCG04], based on the CBR*Tools library and using the *Broadway* approach [TJK99]. Details on the Be-TRIP system are presented in Appendix F while the details on the CBR*Tools integration of our 2-3 AHC algorithm are given in Section 4.3.

  To validate the use of the 2-3 AHC algorithms in the mobility context, some tests were performed on a small generated dataset of urban itineraries in Section 3.5.2.

Part II: Study of applying the 2-3 AHC in the Web Mining and Document Clustering Fields

In the second part of the thesis we will present two first studies of evaluating the interest of applying the Agglomerative 2-3 Hierarchical Classification algorithm in Web Mining and Document Clustering fields.

- **Chapter 5: Web Usage Mining Application**

  The first study consists of applying our 2-3 AHC algorithm to cluster INRIA's research teams visited topics based on the Web users behaviours.

  The goal of the application was to analyze the impact of the site structure (the research themes especially) on the Web users navigations.

  The analysis was performed on two time periods, before and after the research teams reorganization into research themes.

*`http://www-sop.inria.fr/mobivip/`

Our results show the impact of the site structure on users navigations, whilst this study is, according to our knowledge, the first one to apply a hierarchical classification method on Web Usage data.

- **Chapter 6: XML Documents Clustering Application**

  In this Chapter, we applied our 2-3 AHC algorithm on homogeneous XML documents issued from the 2003 activity reports of INRIA's research teams. The goal of the application was the validation of an existing organizational structure using our classification. Our second objective was to compare different 2-3 AHC algorithms using as "reference" the classical AHC one.

  Comparing the semi-structured XML documents is based on their structure only or their structure and content. The following hypothesis is tested here: different parts of the analyzed XML documents correspond to different dimensions of the document collection and might play different roles in the performed classification.

  We found that the best results are obtained with the 2-3 AHC algorithm avoiding the blind merging (V3), which was the only one to always have a positive Stress gain compared to the classical AHC.

Conclusions and Perspectives

Finally in this last Chapter of the thesis document, we will present the contributions, the interests and the limitations of our work. Then, we will point out the future work perspectives related to our study of the 2-3 AHC in the end of this thesis.

Part I

Contributions to Hierarchical Clustering

Chapter 2

Theoretical Study of the Agglomerative 2-3 Hierarchical Classification

Being the newest extension of the classical hierarchies, the 2-3 hierarchies [Ber02d] have been less studied. In this context we begin with a theoretical study of this type of structure, study closely related to the initial 2-3 AHC algorithm from [Ber02d].

In order to present the 2-3 Agglomerative Hierarchical Classification [Ber02d], we first need to present the classical Agglomerative Hierarchical Classification method and its algorithm [SS63] (see Section 2.1) introduced in 1963 by Sokal and Sneath. Based on its concepts and definitions, we will then present in Section 2.2 the 2-3 *hierarchy* concept and the initial 2-3 AHC pseudo-algorithm proposed in 2002 by P. Bertrand [Ber02d]. As we said before, the interest of the 2-3 hierarchies is that they generalize the classical hierarchies and have a richer structure. As in the case of the other extensions of the classical hierarchies (see. Section 1.7), the 2-3 hierarchies allow clusters to overlap.

The theoretical study that we carried out on the Agglomerative 2-3 Hierarchical Classification in this Chapter has revealed four properties (cf. Section 2.3). These properties are used to analyze the algorithm execution influence on the created 2-3 hierarchy (cf. Section 2.6) and then later to propose a new 2-3 AHC algorithm (cf. Chapter 3).

Next we studied different ways of defining an aggregation index (cf. Section 2.4) which is the cluster dissimilarity measure used as an input in the classical AHC and 2-3 AHC algorithms. We thus decided which definition of the aggregation index (called also link) would be more appropriate in the context of the 2-3 hierarchies especially for the complete and single link. Being closely related to the aggregation index, we studied in Section 2.5 the 2-3 hierarchies indexing. A special type of merging called the *blind merging* is revealed (cf. Section 2.6) and analyzed (cf. Section 2.7).

Before concluding in Section 2.9, we first analyze (cf. Section 2.8) which one of the complete link definitions is better from the complexity execution point of view.

Next, in Chapter 3 and based on the study presented in this Chapter, we will first propose a new 2-3 AHC pseudo-algorithm followed by a proposition and first implementation of a 2-3 AHC detailed algorithm version. The advantages of this new algorithm are its reduced complexity (from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2\, log\, n)$) and a principle similar to the classical AHC algorithm. Another variants of the 2-3 AHC algorithm are also proposed to integrate the refinement step into the merging step and to avoid a special merging case, denoted *blind merging*.

In the rest of the thesis we will denote the Agglomerative 2-3 Hierarchical Classification as the 2-3 AHC, whilst the classical Agglomerative Hierarchical Classification will be denoted as the classical AHC, or simply the AHC.

2.1 Classical AHC: Method and Algorithm

We begin by introducing some basic notations that will be used in the rest of the document.

We denote as E the set of objects that are to be analyzed and we assume that there is a total of n objects in this set: $E = \{x_i : i = \overline{1,n}\}$. Each object x_i from this set is described by m variables: $x_i = (x_i^1, x_i^2, ..., x_i^m)$ where $x_i \in E$ and $i = \overline{1,n}$. We suppose that the analyzed set E is described by a *dissimilarity*, say δ, where $\delta(x, y)$ indicates the degree of dissimilarity between two arbitrary objects x and y of E.

Sokal and Sneath proposed in 1963 in [SS63], a very general algorithm which can be considered as the first version of the well known AHC algorithm. It is a very simple classification algorithm based on a distance/dissimilarity matrix between initial elements, denoted δ.

For example, the initial distance measure between the initial elements may be the *Euclidean distance*:

$$d_e(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{2.1}$$

or any other distance such as the *rectangular distance*, the *Minkowski metric*, the *chi-2 distance*, etc. Choosing the distance in also influenced by the type of variables that characterize the individuals (numerical, nominal, symbolic, etc.).

A set of elements is called a *class* or *cluster* and can contain at least one element and at most all the elements.

Besides this distance/dissimilarity measure on the initial elements, the AHC algorithm uses a similarity measure between clusters and a measure of the heterogeneity degree of these clusters. The former is known as the *aggregation index*, *merging index*,

cluster dissimilarity measure or simply *link*, is based on the initial elements distance and is a symmetric mapping:

$$\mu : \mathcal{P}(E) \times \mathcal{P}(E) \rightarrow [0, \infty), \quad \text{with} \quad \mu(\{x\}, \{y\}) = \delta(x, y) \ \forall x, y \in E \tag{2.2}$$

The later is called the *indexing measure* or the *indexing function* and assigns to each cluster an real positive value corresponding to its heterogeneity degree.

The AHC algorithm consists of an *initialization* phase, a recursive *regrouping* (*merging*) phase, and an *ending condition*. In the initialization phase, the initial elements to be analyzed are put into single-elements clusters, called the *singletons*. Their indexing value f (representing the heterogeneity degree), is then set to 0 for all of them: $f(\{x\}) = 0, \forall\, x \in E$.

Next, the recursive regrouping/merging phase is performed. Basically, at each step the two nearest elements in the sense of the chosen aggregation index, are selected from the distance/dissimilarity matrix and regrouped into a single cluster. Then the dissimilarities μ between the new cluster and the rest of the clusters are computed using one of the following aggregation indexes:

- *single linkage* - the closest neighbor:

$$\mu_{sl}(X, Y) = \min\{\delta(x, y) : x \in X, y \in Y\} \tag{2.3}$$

- *complete linkage* - farthest neighbor, (also known as the *total linkage*):

$$\mu_{cl}(X, Y) = \max\{\delta(x, y) : x \in X, y \in Y\} \tag{2.4}$$

- *average linkage* - average distance between elements:

$$\mu_{al}(X, Y) = \frac{\sum \delta(x, y)}{|X| \times |Y|}, \quad \text{with} \quad x \in X, y \in Y \tag{2.5}$$

- and when possible to compute (spaces with numerical dimensions):
  - the distance between *gravity centers*,
  - the *Ward's criterion* [War63]:

$$\mu_{wl}(X, Y) = \frac{|X| \times |Y|}{|X| + |Y|} \times \sum \delta^2(\overline{X}, \overline{Y}) \quad \text{with} \quad x \in X, y \in Y \quad \text{and} \quad \overline{X}, \overline{Y} \text{ the (gravity) centers of } X \text{ and } Y \tag{2.6}$$

  - the weighted pair-group method using averages (WPGMA) [SS73a], etc.

The choice of the distance (dissimilarity) measure between clusters has a big influence on the resulting structure. The average-link, complete-link, and Ward's methods tend to favor spherical clusters, while single-link clustering resembles to density based methods and can produce "elongated" clusters [Fas99].

Generally speaking for all algorithms, the main issue is the computation of these formulas during their execution. Indeed, for each created cluster $X \cup Y$ one must compute the link between him and every other existing cluster Z that can be still merged. The single link and the complete link use the minimum and the maximum between $\mu(X, Z)$ and $\mu(Y, Z)$, but for others the computation could be a problem. But the other links can also be easily computed using different combinations of the existing link values $\mu(X, Y)$ and $\mu(X, Z)$.

For this, Lance and Williams introduced in 1967 [LW67] a general formula:

$$\mu(X \cup Y, Z) = \alpha_x \mu(X, Z) + \alpha_y \mu(Y, Z) + \beta \mu(X, Y) + \gamma |\mu(X, Z) - \mu(Y, Z)| \qquad (2.7)$$

Table 2.1 bellow presents the parameters values for the most common linkages, where n_x, n_y and n_z are the cardinals of the sets X, Y and Z.

Coefficients / **Link**	$\alpha_{\boldsymbol{x}}$	$\alpha_{\boldsymbol{y}}$	β	γ
Single	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
UPGMA (group average)	$\frac{n_x}{n_x+n_y}$	$\frac{n_y}{n_x+n_y}$	0	0
WPGME (weighted average)	$\frac{1}{2}$	$\frac{1}{2}$	0	0
UPGMC (unweighted centroid)	$\frac{n_x}{n_x+n_y}$	$\frac{n_y}{n_x+n_y}$	$-\frac{n_x \times n_y}{(n_x+n_y)^2}$	0
WPGMC (weighted centroid)	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Ward's criterion	$\frac{n_x+n_z}{n_x+n_y+n_z}$	$\frac{n_y+n_z}{n_x+n_y+n_z}$	$-\frac{n_z}{n_x+n_y+n_z}$	0

Table 2.1: Coefficients values for different links

When two clusters are merged, the dissimilarity between them is associated to the new cluster and it represents the "degree" (dissimilarity) at which the elements in the two clusters were regrouped (the heterogeneity degree, f). Usually we have: $f(X \cup Y) = \mu(X, Y)$. Also the relations between the new cluster and the merged ones, are stored for each of them.

The algorithm stops when we have merged the last two remaining clusters and the resulting cluster thus contains all the elements of E.

The result is sequence of nested partitions, that can be visualized using a graphic, called *dendrogram* or *classification tree* (see Figure 2.1).

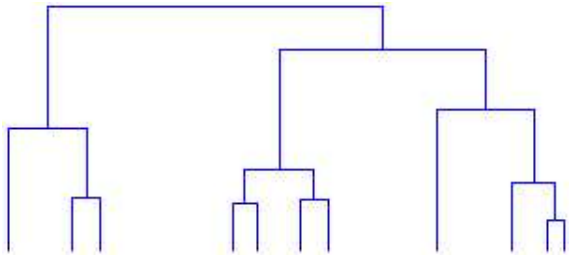


Figure 2.1: Dendrogram

This indexed hierarchy will induce a new distance matrix which is an *ultrametric* over the initial elements, based on the distances at which they were first regrouped. For example, the induced distance δ' between two elements x and y has the value of the indexing function f of the cluster in which they are first regrouped together:

$$\delta'(x,y) = f(A) \quad \text{where} \quad x,y \in A \quad \text{and} \quad \nexists B \subset A \quad \text{such that} \quad x,y \in B. \tag{2.8}$$

The induced ultrametric can be then compared to the initial distance/dissimilarity matrix for quality analysis, by using different indices like the *Stress* formula [JW82], the *Pearson* correlation coefficient, etc.

Information from the indexed hierarchy and its nested partitions can also be used to generate clusterings of the initial elements. These clusterings are actually the nested partitions that form the hierarchy. To select a partitioning, one can chose an indexing level and then select in the hierarchy all the maximal clusters below that level. The level is selected by analyzing the indexing levels of the clusters in an hierarchy to detect if there are any important gaps (distances) between them. As the indexing level represents the cluster heterogeneity degree, it means that the partitioning corresponding to the lower cluster level is more homogeneous. A trade off between the approximate number of desired clusters in a partition and the heterogeneity degree (the clusters levels) must be made however. This partitioning technique is sometimes referred as "cutting" the hierarchy.

The dendrogram can be also analyzed from the singletons proximity (grouping) point of view. This kind of analysis is especially used in bioinformatics by the genome researchers who are trying to find meaningful clusters in DNA microarray data, also known as gene arrays or gene chips. Finding clusters of genes with similar expression

patterns could lead to better understanding of the functions those genes. One of the most studied cases is the human genom, which contains approximately 40,000 genes. The dendrogram can be horizontally displayed (see Figure 2.2 presenting a small hierarchy of INRIA's research teams), while individuals are visually depicted (colors, labels) to ease the interpretation: Hierarchical Clustering Explorer [SS02], dChip [LW03], CAP [ADLCR99], Bioinfomatics toolbox [Mat].

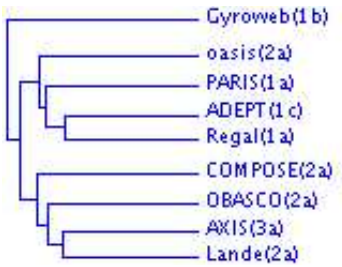


Figure 2.2: Horizontal dendrogram of a hierarchy representation

One of the main characteristics of the hierarchies, produced by the AHC algorithm, is the fact that once grouped, the elements rest together all the time. We will see that this is the main difference between hierarchies and the 2-3 hierarchies introduced in [Ber02d], which allows the clusters to intersect each other and use a slightly modified aggregation indexes.

2.1.1 Definitions and Properties

We start with definitions from a general setting point of view that extends the framework of hierarchies.

A *set system* on E is a nonempty collection of nonempty subsets of E having E as one of its elements. We consider a collection $\mathcal{C}$ of nonempty subsets of E, often called *clusters* in the rest of the text. We will say that the collection $\mathcal{C}$ is *hierarchical* if $X \cap Y \in \{\emptyset, X, Y\}$, $\forall X, Y \in \mathcal{C}$ (see also Definition 1). Let us recall that a *hierarchy* on E is a hierarchical collection which contains E and its singletons.

Definition 6 : *A* **successor** *of $X \in \mathcal{C}$ is any maximum element of $\{Y \in \mathcal{C} : Y \subset X\}$ ordered by the set inclusion order. If Y is a successor of X, then X is said to be a* **predecessor** *of Y.* □

The set of all successors (resp. predecessors) of $X \in \mathcal{C}$ is noted $succ(X)$ (resp. $pred(X)$). If $\mathcal{F} \subseteq \mathcal{C}$ is also a collection of non-empty subsets of E, then $succ(\mathcal{F})$ (resp. $pred(\mathcal{F})$) represents the collection of clusters that are successors (resp. predecessors) of at least a set from $\mathcal{F}$.

Definition 7 : *A collection $\mathcal{C}$ is said to be* **pre-indexed** *if there is a isotone mapping $f : \mathcal{C} \to \mathbf{R}^+$ such that f is defined $\forall\, x \in \mathcal{C}$ and:*

- for all $X, Y \in \mathcal{C}$ with $X \subset Y$ we have $f(X) \leq f(Y)$. □

When f is strict (meaning f strict *isotone*) the collection $\mathcal{C}$ is said to be *strictly indexed*:

- for all $X, Y \in \mathcal{C}$ with $X \subset Y$ we have $f(X) < f(Y)$.

Definition 8 : *The collection $\mathcal{C}$ is* **weakly indexed** *by a map $f : \mathcal{C} \to \mathbf{R}^+$, if $X \subset Y$ implies $f(X) \leq f(Y)$ and if $f(X) = f(Y)$ with $X \subset Y$, implies that X is equal to the intersection of its predecessors.* □

In the following, expressions as "*maximal cluster*" and "*noncomparable clusters*" will be used in the sense of the set inclusion order.

2.1.2 Examples

A small example of a dissimilarity matrix on seven individuals and used as input for the classical AHC algorithm is presented in Table 2.3. Given n individuals (initial elements) to cluster, the size of the input matrix is $\frac{n(n-1)}{2}$. This can be difficult to manipulate for larger n.

	a	b	c	d	e	f
b	2					
c	3	3				
d	5	4	9			
e	6	5	8	1		
f	4	7	7	1	1	
g	8	9	7	8	5	6

Figure 2.3: Dissimilarity matrix example

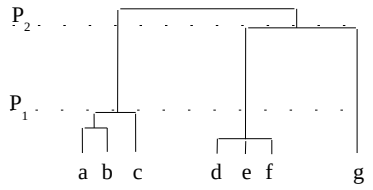


Figure 2.4: Hierarchy partitioning

Using the complete-linkage, we obtain the indexed classical hierarchy from Figure 2.4. Here a and b are successors of $\{ab\}$ while $\{ab\}$ is their predecessor for example. As we mentioned before, in order to chose a partition from the resulting hierarchy, the differences between the f level of the created clusters are analyzed. Usually the partitioning level is chosen whenever there's a "*big*" difference between the created cluster's levels. For example in Figure 2.4, the first partition $\{\{a, b, c\}, \{d, e, f\}, \{g\}\}$ generated by the level $P_1 = 3$, is more appropriate than the second one $\{\{a, b, c\}, \{d, e, f, g\}\}$ generated by $P_2 = 8$, since the clusters are more homogeneous.

2.2 Agglomerative 2-3 Hierarchical Classification (2-3 AHC)

In this section we present the 2-3 *hierarchy* concept along with the 2-3 *Agglomerative Hierarchical Classification* method introduced both in [Ber02d] in order to generalize and to make more flexible the classical AHC.

As we saw before, the AHC generates disjoint clusters or clusters included one in the other. The *2-3 Agglomerative Hierarchical Classification* method gives each cluster the possibility of intersecting at most another cluster, when the obtained intersection is distinct from the two clusters (see Figure 2.5).

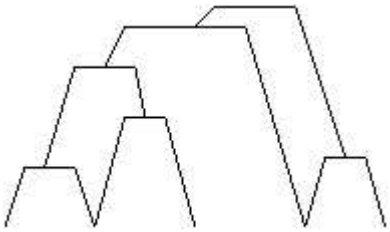


Figure 2.5: Example of a 2-3 hierarchy

This characteristic allows the obtained cluster structure to highlight groups of objects having the common characteristics of two other groups (not possible with the AHC).

The resulting cluster structure is called an 2-3 *hierarchy*, term justified by the following property equivalent with the aforementioned characteristic:

Property: `Given any three clusters, at least two out of the three possible clusters pairs are hierarchical (nested or disjoint).`

The term *2-3 hierarchy* specifies how the set of 2-3 hierarchies is an extension of the hierarchies set - indeed, from the definition above it clearly results that a hierarchy is a particular case of 2-3 hierarchy. This happens when all three possible cluster pairs are hierarchical, thus leading to a classical hierarchy.

2.2.1 Definitions and Properties

In the following, E will designate a nonempty finite set of size n, and $\mathcal{C}$ will be a collection of nonempty subsets of E.

Denoting X and Y as two sets, we will say that a subset Y of X is *proper* when it's both nonempty and distinct from X. Also Y is a *trivial subset* of X if Y is not proper or reduced to some singleton of X.

Definition 9 : *If two subsets X and Y of E satisfy $X \cap Y \notin \{\emptyset, X, Y\}$, or in other words if the pair $\{X, Y\}$ is not hierarchical, then we will say that* ***X properly intersect Y****, denoted $Xp \cap Y$.* □

For example in Figure 2.6, X properly intersects Y and $Xp \cap Y = Z$.

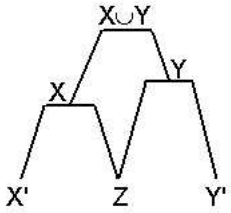


Figure 2.6: Example of a *proper intersection*: $Xp \cap Y = Z$

We present now the 2-3 hierarchy definition as given in [Ber02d] (equivalent to Definition 5):

Definition 10 : *A* **2-3 hierarchy** *on E is a collection $\mathcal{C}$ that:*

- *contains E and its singletons,*
- *is closed under nonempty intersections,*
- *and such that each element of $\mathcal{C}$ properly intersects no more than one other element of $\mathcal{C}$.*

□

Also, the following concept of *2-3 hierarchical* collection was introduced by P. Bertrand in [Ber02d]:

Definition 11 : *A collection $\mathcal{C}$ of subsets of E is called 2-3 hierarchical if each element of $\mathcal{C}$ properly intersect no more than one other element of $\mathcal{C}$, or in other words if for all $X \in \mathcal{C}$, we have $\mid \{Y \in \mathcal{C} : X \cap Y \notin \{\emptyset, X, Y\}\} \mid \leq 1$.* □

Two main properties of this type of collection are: first, a 2-3 hierarchy on E is a family of intervals of at least a linear order defined on E (cf. Theorem 4.6 in [Ber02d]). This property allows to represent graphically a 2-3 hierarchy as a pyramidal classification (cf. the Figure 2.5). Secondly, according to Theorem 3.3 in [Ber02d], any

2-3 hierarchy on E has a maximum number of elements of $\lfloor \frac{3}{2}(n-1) \rfloor$, excluding the singletons of E.

The main advantage of the 2-3 hierarchies compared to the classical hierarchies is their richer structure. Indeed, since the 2-3 hierarchies allow clusters to properly intersect themselves, their structures are richer compared with the classical hierarchies obtained on same datasets [CBT04]. For example the maximal number of created clusters by the classical AHC is $(n-1)$, compared with $[\frac{3}{2}(n-1)]$ for the 2-3 AHC [Ber02d], where n is the initial number of elements.

This information gain can also be noticed in the case of the properly intersecting clusters: we know which one of the successors made the merging possible. Or in other words, we can see which part of a cluster is closer to the other cluster. For example in Figure 2.6 we can say that Z is closer to Y' than to X'.

As concerning the aggregation index, it is clearly that we need new definitions for the special case of properly intersecting clusters. We can thus use the *single linkage* distance, as defined in [Ber02d], to compute the dissimilarity between two sets that might intersects each other or not.

$$\mu_{sl}(X,Y) = \min\{\delta(x,y) : x \in X - Y, y \in Y - X\} \tag{2.9}$$

For the *complete linkage*, the dissimilarity between two sets is the same as in the case of the AHC (formule 2.4) even for the clusters that are not disjoint:

$$\mu_{cl}(X,Y) = \max\{\delta(x,y) : x \in X, y \in Y\} \tag{2.10}$$

Also the indexing formula will be changed for the 2-3 hierarchical case, to take into account the properly intersecting clusters and the cluster inversions (see below). Thus, the chosen aggregation index is used to determine f, the degree of heterogeneity inside a newly formed cluster, with the next formula:

$$f(X \cup Y) = max\{f(X), f(Y), \mu(X,Y)\} \tag{2.11}$$

In the rest of the text we will refer to this formula as the *extended indexing formula*, while $f(X \cup Y) = \mu(X,Y)$ will be called the *normal indexing formula*.

For the *single link*, an extension of the normal indexing formula was proposed in [Jul02a]. It was called the *double single-link* formula, and was introduced in order to avoid inversions in the resulting 2-3 hierarchy for the single-link when using the afore mentioned indexing formula. An inversion (see Figure 2.7 and Section 2.4.2.2) appears when a successor of a cluster has a bigger indexing value f, than the one of the cluster itself:

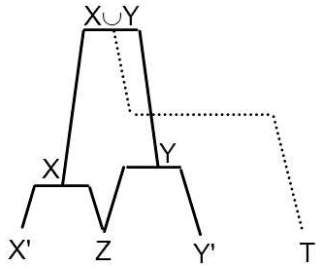


Figure 2.7: Example of levels inversion

$$f(X \cup Y) > f(X \cup Y \cup T) = \mu(X \cup Y, T).$$

For this, the normal indexing formula was slightly altered in order to have a pre-indexed 2-3 hierarchy. This new indexing formula is called the *double single-link* and is defined as follows (refer to the example from Figure 2.7 for notations):

$$\begin{aligned} f(X \cup Y) &= \mu(X, Y) \text{ for disjoint clusters,} \\ f(X \cup Y) &= min\{\mu(X', Y'), \mu(X \cup Y, T)\} \text{ where } T \text{ is the closest candidate} \\ &\quad \text{cluster to } X \cup Y, \text{ for clusters that properly intersects} \\ &\quad \text{themselves.} \end{aligned} \tag{2.12}$$

This is illustrated on the example from Figure 2.8 bellow.

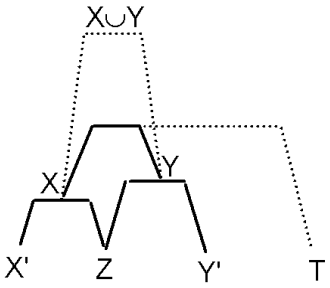


Figure 2.8: Using the *double single-link*

The inversions are sometimes refered as crossovers or reversals [JD98].

2.2.2 Examples

A small 2-3 hierarchy example is presented in Figure 2.9 bellow. Here the cluster $\{ba\}$ *properly intersects* $\{ac\}$, while the clusters $\{ba\}$, $\{bac\}$ and $\{de\}$ are hierarchical when compared one to each other.

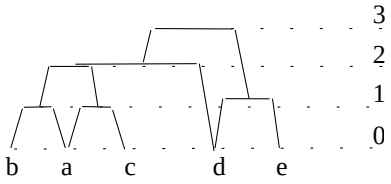


Figure 2.9: Example of an 2-3 Hierarchy

As we mentioned before, the 2-3 hierarchies are richer than the classical hierarchies obtained on same datasets. This can be observed for example on the number of created clusters by the two methods. Figure 2.10 illustrates this by a small example of created hierarchy and 2-3 hierarchy on a three points dataset.

	b	c
a	1	2
b		3

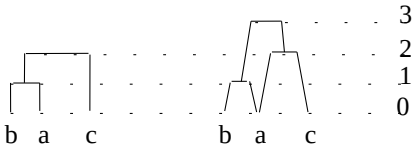


Figure 2.10: AHC and 2-3 AHC

As it can be clearly seen, the 2-3 hierarchy contains more clusters and it better represents the initial dataset compared to the classical hierarchy.

2.2.2.1 Example of Levels Inversion

We present here an example of a small dataset (Figure 2.11) and the created 2-3 hierarchy using the single-link (cf. Figure 2.12) and the complete-link (cf. Figure 2.14). The normal indexing formula is used.

The dataset is a set of four points with the distance matrix presented in Figure 2.11.

	b	c	d
a	1	5	3
b		2	4
c			3

Figure 2.11: Dataset

To avoid such levels inversions (cf. Figures 2.12 and 2.14), one can use for the single-link the *double indexing formula* (cf. Figure 2.13) whilst for the complete-link, the *extended indexing formula* (cf. Figure 2.15).

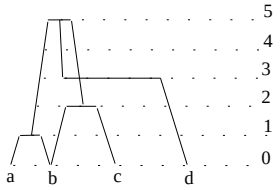


Figure 2.12: Single-link levels inversion

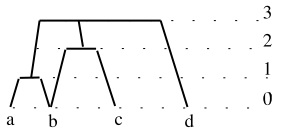


Figure 2.13: The *double single link*

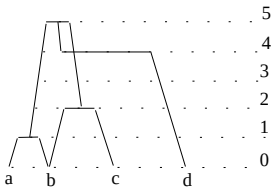


Figure 2.14: Complete-link levels inversion

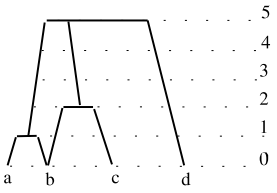


Figure 2.15: *Extended indexing formula*

2.2.3 2-3 AHC Initial Algorithm

In this subsection we present the first 2-3 AHC algorithm proposed in [Ber02d] in order to generalize the classical AHC algorithm. This algorithm has a $\mathcal{O}(n^3)$ time complexity (see Prop. 5.5 in [Ber02d]), and it was not implemented.

It is this first algorithm that we used as a base for our theoretical study and for the proposal of a new 2-3 AHC algorithm in Chapter 3.

Initial Algorithm of the 2-3 AHC:

1. **Initialization**: $i = 0$; The set of clusters and the set of candidate clusters $\mathcal{M}_i$ coincide with the set of singletons of E; $f(\{x\}) = 0$, $\forall x \in E$.
2. **Merge**: $i = i + 1$; Merge a pair $\{X_i, Y_i\}$ such that $\mu(X_i, Y_i) \leq \mu(X, Y)$, among the pairs $\{X, Y\} \subseteq \mathcal{M}_{i-1}$, which are noncomparable and satisfy α or β :
   (α) X and Y are maximal, and X (resp. Y) is the only cluster susceptible to properly intersect Y (resp. X).
   (β) One of X or Y is maximal, and the other admits a single predecessor Z. No cluster is properly intersected by X, Y or Z.
3. **Update**: $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$, from which we eliminate any cluster strictly included in at least a cluster of $\mathcal{M}_{i-1}$ and in $X_i \cup Y_i$. Update μ by using an extension of Lance and Williams Formula. Update f by using $f(X_i \cup Y_i) = \max\{f(X_i), f(X_i), \mu(X_i, Y_i)\}$.
4. **Stop test**: repeat steps 2 et 3, until the cluster E is created.
5. **Refinement**: remove some clusters so that f is a weak index.

We briefly present this algorithm here, additional details and the detailed version of this algorithm from [Ber02d] can be found in Section 3.1.

As we can see, the algorithm consist as in the classical AHC case, in a initialization phase, a recursive merging and update phase and an ending condition.

The main difference is that here are two types of merging, denoted the α and β mergings and which involve also clusters that properly intersects themselves. Also, the set of clusters to be eliminated from the candidates clusters $\mathcal{M}_i$ (the clusters eligibles to be merged with other clusters) is reduced here compared to the classical case. This actually lets non-maximal clusters to be merged with other clusters, creating thus the 2-3 hierarchical structure.

In step `3` the set of the created structure so far and $\mathcal{M}_i$ are updated and the dissimilarities between the new clusters and the rest of the clusters in $\mathcal{M}_i$ are computed according to the dissimilarity measure [Ber86] chosen in the beginning. For the single linkage, formula (2.9) is used to compute the dissimilarity between two clusters.

Another important aspect regarding the two clusters chosen for merging is their cardinality in case of multiple matches:

(γ) when we have multiple pairs of clusters satisfying the minimum dissimilarity condition, we will merge the clusters having the minimal cardinality.

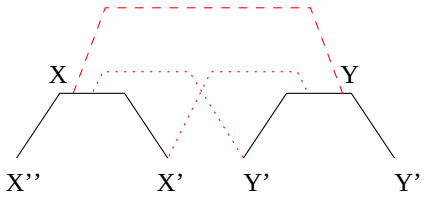


Figure 2.16: The γ condition

For example in Figure 2.16, one of the pairs (X', Y) or (X, Y') will be the one merged, and not the pair (X, Y). This simple condition actually allows clusters to properly intersect themselves, by favorizing mergings between maximal and non-maximal clusters. If the maximal cardinality clusters (X and Y) would be merged in this case, the algorithm will construct a classic hierarchy instead.

In the refinement step the created structure is analyzed in a top-down manner. Starting from the last formed cluster (the E set), the indexing f value is compared with the ones of its successors. In case of equality the successor is removed (it does not present any interest from the clustering point of view) excepting the case were it's also included in another cluster. Then the connections $(succ, pred)$ in the 2-3 hierarchy are "redirected" in order to maintain a 2-3 hierarchical structure. Normally after this update in the structure of the 2-3 hierarchy, the test is repeated for the new successors

of the cluster in order to obtain a weakly indexed structure.

The interest in obtaining such a mapping, apart from the mathematical consequences, is obvious: the "lecture" of the weakly indexed 2-3 hierarchy will be simpler than the case where f is a mapping in a large sense and also because of its reduced number of clusters.

It has been proved by P. Bertrand in [Ber02d] that the algorithm presented before performs in $\mathcal{O}(n^3)$ (Proposition 5.5).

Just like in the case of the hierarchies, an induced dissimilarity matrix can be obtained from the indexed 2-3 hierarchies.

Definition 12 : *A dissimilarity δ on E is called a* **2-3 ultrametric** *if for each four-element subset X of E, there exists a non trivial subset Y of X such that:*

$$\forall x \in X - Y \text{ and } \quad \forall y, y' \in Y, \quad \delta(y, y') \leq \delta(x, y) = \delta(a, y').$$

□

Indeed, this correspondence is bijective as shown by theorem 6.23 of [Ber02d].

2.3 Four New 2-3 AHC Properties

In this section we present our theoretical study of the 2-3 AHC method. As follows we will make different assumptions but without any loss of generality for each of them.

2.3.1 The case of two clusters that properly intersect

A first property concerns the maximal size of a 2-3 hierarchy :

Property 1: The maximal size of a 2-3 hierarchy defined on a nonempty set E is $[\frac{5|E|-3}{2}]$.

Proof: This is a trivial consequence of Theorem 3.1 from [Ber02d] (the max size of the 2-3 hierarchy without the singletons plus the number of singletons).

This property will be used later during the complexity analysis of our new 2-3 AHC algorithm in Section 3.4.

A particular issue for the suite of our reasoning is the case of two clusters that *properly intersect* each other as they represent the principal characteristic of a 2-3 hierarchy and the way we handle them can significantly change the outcome of the 2-3 hierarchy construction algorithm.

As mentioned by Laurent Jullien in [Jul02b], a first problem arises when we have to choose the two subsets X_i and Y_i for merging in the case of a *proper intersection*. When we have two clusters that *properly intersect* each other say X and Y, like the ones in Figure 2.17, there are three candidate pairs for merging: (X, Y), (X', Y), (X, Y') with the same cluster as result of the merging: $X' \cup Y' \cup Z'$ and we can have different results according to the dissimilarity measure chosen: *single linkage*, *complete linkage* or other.

In the following, we consider the *single linkage* case and we use the fact that the dissimilarity between two disjoint clusters is less than or equal to the dissimilarity between one of them and a successor of the other. For the *complete-linkage* all the three pairs from Figure 2.17 would respect the minimum distance requirement and thus (X', Y) and (X, Y') would be preferred according to γ. The f value will be the same in all tree cases and it would not influence the refinement step but the created structure would contain *orphan* clusters (see below).

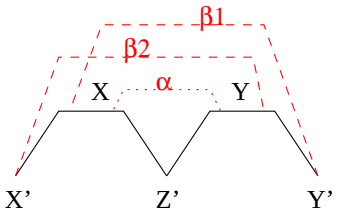


Figure 2.17: α and β merging when X and Y *properly intersect* each other

We assume that Y was the last formed (after X) and in this case we have $\mu(X', Z') \leq \mu(Z', Y')$, otherwise Y would be formed before X and we assume that $\mu(Z', Y') \leq \mu(X', Y')$, otherwise Y would be $\{X' \cup Y'\}$. It's worth mentioning that the clusters X', Y' and Z' are disjoint, thus respecting the 2-3 hierarchy definition.

There are three pairs candidates for merging: (X, Y), (X', Y) and (X, Y') and we will analyze the value of f for the formed cluster in each situation, using (2.9) for the dissimilarity between their elements, even though the selected pair in this case would be (X', Y) (it has the smallest dissimilarity: $\mu(X', Y) = \mu(X', Z')$ according to the *single-linkage*).

In the case of the (X, Y) pair we have:

$f_\alpha = max\{f(X), f(Y), \mu(X, Y)\} = max\{f(X), f(Y), \mu(X', Y')\}$

For the pair (X, Y') we have the following situation:

$f_{\beta 1} = max\{f(X), f(Y'), \mu(X, Y')\} \leq max\{f(X), f(Y'), \mu(Z', Y')\} \leq$

$\leq max\{f(X), f(Y)\}$, by using (2.9) and the definition of f.

And for the (X', Y) pair we have:

$f_{\beta 2} \leq max\{f(X), f(Y)$, using a similar proof.

We can deduce that the α merging will result in a richer 2-3 hierarchy for the *single link*, since the value of f for the formed cluster will be also influenced by $\mu(X', Y')$. If

we use the β merging, the indexing value of the resulting cluster will be either $f(X)$ or $f(Y)$. Thus, in the refinement step one of the successors X, Y of the formed cluster will be eliminated since they are not a proper intersection and the result will be simple hierarchy.

Clearly $f_{\beta 1} \leq f_\alpha$ and $f_{\beta 2} \leq f_\alpha$ where X', Y', Z' are disjoint (otherwise would contradict the 2-3 hierarchy definition) and so the β regrouping will tend to create clusters with the same value as one of their successors causing the successors to be eliminated from the 2-3 hierarchy.

A example on a small dataset (see Table 2.2) is presented in Figure 2.18 (the α case) and in Figure 2.19 (the β case).

	a	b	c	d	e
b	1				
c	2	1			
d	2	1	2		
e	2	2	2	1	
f	2	2	1	2	1

Table 2.2: Dissimilarity matrix example for α and β case

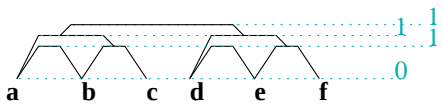


Figure 2.18: f value for β case

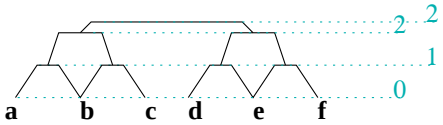


Figure 2.19: f value for α case

Another argument for using the α merging over the β one, is to avoid the formation of *orphan* clusters. This appears when clusters inside the 2-3 hierarchy have no predecessors (except for the last created cluster). For example in Figure 2.17, when using the β_1 (respectively β_2) merging, the cluster Y (resp. X) cannot be merged another cluster since is already merged with X (resp. Y), and thus it will have no predecessor.

As follows we will not try to use the β merging in these conditions and we will exclude this situation using only the α merging in this case. If the β merging were used, the hierarchy created before the refinement step would have almost the same structure, most probably with different values for the mapping f and would contain *orphan* clusters.

2.3.2 Candidate Clusters Elimination

Another property of the 2-3 hierarchy concerns the clusters to be eliminated from $\mathcal{M}_i$ at the end of each merging step in the 2-3 AHC algorithm (cf. Section 2.2.3). The purpose is to specify the exact clusters to be eliminated from $\mathcal{M}_i$, and to avoid to

iterate over this set in order to find the clusters to eliminate. This will help reduce the `Update` step's complexity of the algorithm.

The basic idea is to analyze and possibly eliminate only the first level successors, not the whole $\mathcal{M}_i$ set.

Property 2: The update of $\mathcal{M}_i$ in the initial 2-3 AHC algorithm can be reduced to: Eliminate any successors of X_i and Y_i and even one of them self if it is a common successor for two maximal clusters (in other words if X_i or Y_i is the result of a *proper intersection* between two maximal clusters).

Proof: This can be very easily proved by induction on the number i. For $i = 1$, no cluster is eliminated and $\mathcal{M}_i$ satisfies the property. Using the induction hypothesis we argue that the others clusters that would be candidates for elimination (the successors of $suc(X_i, Y_i)$ and so on) have already been eliminated in a previous step (when X_i and Y_i were created) and the only ones included in the union $X_i \cup Y_i$ and in another cluster of $\mathcal{M}_i$ are $suc(X_i)$, $suc(Y_i)$ and possibly X_i or Y_i in case one of them has a second predecessor (a *proper intersection* case).

Thus, after this modified Update step, $\mathcal{M}_i$ will contain the following clusters (see also the example in Figure 2.20):

- maximal two-by-two disjoint clusters and their successors ($\{abc\}$, $\{ab\}$, $\{bc\}$, $\{de\}$, $\{d\}$, $\{e\}$ in Figure 2.20);
- maximal clusters that *properly intersect* each other and their successors excepting the common one ($\{fg\}$, $\{f\}$, $\{h\}$ in Figure 2.20).

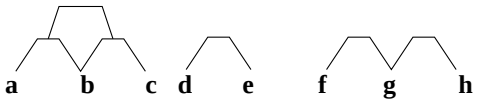


Figure 2.20: Successor candidates elimination

So far these are the candidate clusters for merging at each step of the algorithm, but we will further reduce the set $\mathcal{M}_i$ (see next Section 2.3.3).

2.3.3 Intermediate Merging

As follows we will eliminate from $\mathcal{M}_i$ the pairs of clusters that *properly intersect* each other and their successors. We will do this by operating another merging phase whenever there is a *proper intersection* and by proving that these clusters can't be merged with other clusters in future steps of the algorithm.

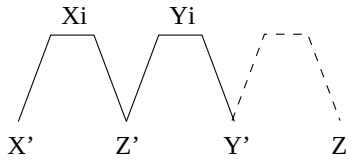


Figure 2.21: Successor case of a *proper intersection*

We first study the successors case of a *proper intersection* (see Figure 2.21) and more specifically the possibility of merging one of these clusters with another cluster disjoint from the properly intersecting ones.

Suppose that we have two clusters X_i and Y_i with a common successor say Z' (cf. Figure 2.21) and with the other successors: X' for X_i and Y' for Y_i. Clearly the last formed of them (say Y_i) was formed at a step k as a result of a β merging between Z' and Y'. Using *single linkage* we have that $\mu(Z', Y') \leq \mu(X_i, Y')$ and if there is equality, according to the γ condition, (Z', Y') will be preferred for merging due to the smaller cardinality of Z'.

We can state the following property here, regarding the successors of X_i and Y_i where X_i and Y_i properly intersect each other (Figure 2.21):

Property 3: If two clusters properly intersect each other then their successors can't be merged in a future step with a cluster disjoint from their union.

Proof: Assume that Y' will be merged with another (non-comparable) cluster Z, disjoint from Y': $Z \cap Y' = \emptyset$. If Z and Y' would properly intersect each other, then the 2-3 hierarchy would not be respected since Y_i would properly intersect both X_i and Z. In this case, Y_i would also properly intersect both $Y' \cup Z$ and X_i which contradicts the 2-3 hierarchy definition. It results that Y' and Z are disjoint.

Using the same reasoning, the clusters Y' and Z can not be merged since the Y_i would properly intersect both X_i and $Y' \cup Z$. We also know that a successor can't be merged with the other cluster itself (β merging in Figure 2.17). Thus, the successors X' and Y' can not be merged with other clusters in a future step of the algorithm.

Using this, we can eliminate from $\mathcal{M}_i$ at the end of each merging step all the successors of the clusters that properly intersect each other, since they are "useless" (they cannot be merged with other clusters and they will unavoidable be later eliminated from $\mathcal{M}_i$ as soon as their predecessors will merge in a higher level cluster). For example in Figure 2.20, if we suppose that the last formed cluster is $\{gh\}$, then $\{f\}$, $\{g\}$ and $\{h\}$ are no longer candidate clusters and can be eliminated from $\mathcal{M}_i$.

Next, we will prove that also the clusters that properly intersect each other can not be merged with other cluster disjoint from themselves (see Figure 2.22 - the predecessor

case of a proper intersection). This will lead us to conclude that two clusters that properly intersect each other will unavoidably be merged together in a future step.

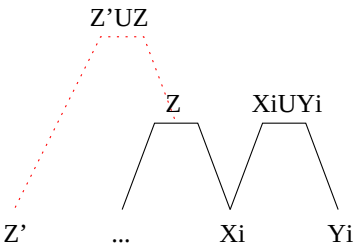


Figure 2.22: Predecessor case of a *proper intersection*

Property 4: If one of the clusters X_i or Y_i chosen for merging at the i step is a successor of a maximal cluster Z from $\mathcal{M}_{i-1}$ then $X_i \cup Y_i$ (resp. Z) can't be merged with a disjoint cluster from Z (resp. $X_i \cup Y_i$).

Proof : Let X_i be the successor of Z that will be merged with Y_i, resulting in a new cluster $X_i \cup Y_i$ which will obviously properly intersect Z (Figure 2.22). Suppose first that Z is merged with $Z' \neq X_i \cup Y_i$. Then $X_i \cup Y_i$ would properly intersect Z and $Z \cup Z'$, which is contradictory. Since the same argument applies when $X_i \cup Y_i$ is merged with $Z' \neq Z$, we conclude that the property holds.

We have supposed that in case of two clusters that properly intersect each other we can't use a β merging between one of them and the disjoint successor of the other cluster.

Using Properties 3 and 4 we can introduce after each merging phase of the algorithm the next test (Figure 2.23):

"If $pred(X_i) = 2$ (or $pred(Y_i) = 2$) then merge the two predecessors of X_i (or Y_i) and remove from $\mathcal{M}_i$ any cluster included in the maximal cluster formed after this union".

We call this new merging *intermediate merging* as it might occur after the merging of two clusters. This seems to be the optimal evolution of the 2-3 AHC algorithm in the case of a proper intersection (see Figure 2.23).

This condition will guarantee that at the end of each step of the algorithm, the set $\mathcal{M}_i$ will contain only maximal disjoint clusters and their disjoint successors (if they are any), while the other clusters that can't be candidates for merging are eliminated.

Also, in three out of four cases (see Section 2.3.4), it won't be necessary to compute the distances between the last formed cluster (Y in Figure 2.23) and the rest of the candidates from $\mathcal{M}_i$, since it would be eliminated from $\mathcal{M}_i$ right after its creation

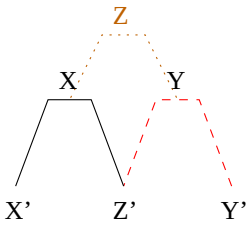


Figure 2.23: Evolution in the common successor case

(the distances between Z and the other candidate clusters can be computed using the distances between X,Y' and the other candidate clusters).

Remark 5: In this context the (α) and (β) conditions on choosing X_i and Y_i from $\mathcal{M}_i$ can be reformulated into a single condition as it follows:

(a) *At most one among X_i, Y_i is successor of another cluster from $\mathcal{M}_i$.*

This properties will be used later in Chapter 3 to reformulate the 2-3 AHC algorithm and to propose a new 2-3 AHC algorithm with a principle similar to the classical AHC algorithm.

2.3.4 Integrating the Refinement Step into the Merging Step

This analysis has as purpose the elimination of the refinement step at the end of the algorithm, which can be very expensive due to its recursive nature [Ber02b]. Instead, the refinement will be performed "on-the-fly" after each merging step during the algorithm execution. The basic idea is to integrate the refinement step into the merging step of the algorithm. We will prove that we obtain a weakly indexed 2-3 hierarchy.

Nevertheless we first must question the refinement itself in the case of the 2-3 hierarchies since it can eliminate important information in the case of clusters that properly intersect. This information concern actually the order of the clusters in the resulting 2-3 hierarchy (see Section 2.2), not their indexing levels and the induced dissimilarity matrix.

Suppose we have three "equidistant" clusters which are merged together (see Figure 2.24). In the created hierarchy, the cluster $\{ab\}$ presents no interest since is on the same level with $\{abc\}$ and there are four possible orders (2^{3-1}) on the initial elements. But in the 2-3 hierarchy, we have only two possible orders plus the information that $\{b\}$ is the "intermediating" cluster. When performing the refinement, these informations will be lost, but they will not influence the induced dissimilarity, only the results interpretation.

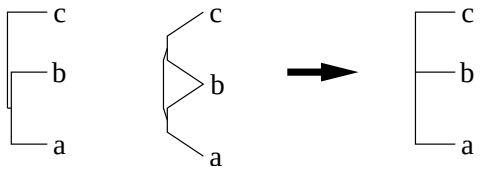


Figure 2.24: Refinement influence

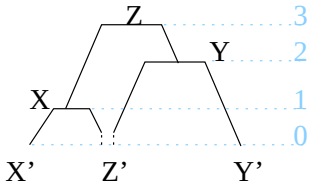
In conclusion, for the 2-3 hierarchies when we are interested in the singletons order is better to avoid the refinement step at the end of the algorithm to keep as much information available. When this order is not relevant for our analysis, one can see that the non-maximal clusters involved in a proper intersection like the one in Figure 2.24 ($\{a\}$, $\{b\}$, $\{c\}$) will no longer be candidate clusters. Meanwhile, $\{ab\}$, $\{bc\}$ will be eliminated in refinement step at the end of the algorithm. So beside the above mentioned reason (avoiding recursive refinement), we can chose to perform an "on-the-fly" refinement and thus allow $\{a\}$, $\{b\}$ and $\{c\}$ to possibly be merged with other candidates. Locally, this would lead to a richer 2-3 hierarchy since theoretically there will be more candidate clusters, but globally the outcome is unpredictable since we influence the way the 2-3 hierarchy is constructed.

In this context, the integrated refinement step from the algorithm(s) that we will later propose (see Chapter 3) will be an optional step depending on the purpose of the analysis.

To integrate the refinement step into the merging step we will perform an exhaustive analysis of the cases concerning the merging value f of a new cluster and the merging values of his two merged successors during the execution of the 2-3 AHC algorithm. The purpose is to eliminate the successors found on the same level with their predecessor without any information lose. There are two possible situations: when there is a *proper intersection* between the successors, respectively when the successors are disjoint. The later contains also two situations: in the first, one of them will have two predecessors after the merging, and in the second both of them will have as unique successor the new formed cluster.

In order to prove that the obtained 2-3 hierarchy will be weakly indexed, we proceed by induction. We remind that a collection $\mathcal{C}$ is weakly indexed if $X \subset Y$ implies $f(X) \leq f(Y)$ and if $f(X) = f(Y)$ with $X \subset Y$, implies that X is equal to the intersection of its predecessors or in other words when X represents a proper intersection. We suppose that until a step i of the algorithm the structure is weakly indexed (true for step 1 when $\mathcal{M}_i$ contains only singletons).

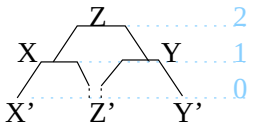
We assume that we have the clusters X and Y to merge at the step $i+1$. Using the formula (2.9) to determine the f value of a new cluster, there are four possible cases:



a.

Figure 2.25: $f(X) < f(Y) < f(Z)$

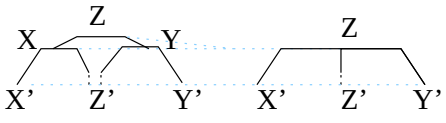
1. $f(X) < f(Y) < f(Z)$: In this case (Figure 2.25), the value of f for X, Y and Z will be all different. The 2-3 hierarchy will be maintained and it will remain weakly indexed. The successors of X and Y, if any, will be eliminated in this case from $\mathcal{M}_i$ (Property 3). If X, Y *properly intersect* each other then they will be eliminated too from $\mathcal{M}_i$ (Property 4). If they don't *properly intersect* then one of them could have a second predecessor, but without any influence on their merging.



b.

Figure 2.26: $f(X) = f(Y) \leq f(Z)$

2. $f(X) = f(Y) \leq f(Z)$: Here (Figure 2.26) we could not eliminate X (or Y) because in this case X' (or Y') and Z' will be regrouped at the distance 2 instead of 1. The structure will be maintained since the formed cluster will have a bigger merging value as his successors (same reasoning like in **1** in case of a *proper intersection* and of two predecessors). The structure will be weakly indexed since $f(X) = f(Y) \leq f(Z)$.



a.

Figure 2.27: $f(X) = f(Y) = f(Z)$

3. $f(X) = f(Y) = f(Z)$: (Figure 2.27). This means that X', Y' and Z' were all regrouped at the same distance, but in different clusters. We will eliminate X and Y from $\mathcal{M}_i$ and $\mathcal{S}_i$ because they don't present any interest here. Thus, X', Y' and Z' will be all directly merged into Z. This is the case of the *equidistant* clusters. In case there

is no *proper intersection*, Z' is not common, X (or Y) can have two predecessors and in this case Z will replace X (or Y), and his successors will be eliminated from $\mathcal{M}_i$. As for the weak indexing, we have two cases: first $f(Z') < f(X) = f(Y)$ and in this case the weak indexing will be preserved. Secondly, if $f(Z') = f(X) = f(Y)$, it means that Z' was necessarily the proper intersection of X and Y. In this case we can eliminate Z' from the structure since it does not properly intersect another cluster and since it will be on the same level as it's new predecessor, Z ($f(Z') = f(X) = f(Y) = f(Z)$). Thus the structure will remain weakly indexed.

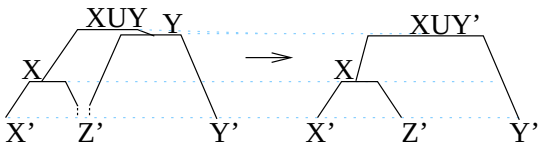


Figure 2.28: $f(X) < f(Y) = f(Z)$

4. $f(X) < f(Y) = f(Z)$: (Figure 2.28). We have two cases here: Z' is the common intersection of X and Y, or X and Y do not properly intersect themselves and one of them could have another predecessor. In the first case, Y can be eliminated from $\mathcal{M}_i$ and $\mathcal{S}_i$. Also $succ(X)$ will be removed from $\mathcal{M}_i$ (Property 4).

If X and Y do not *properly intersect* each other then X (or Y) can have two predecessors. Here we have two situation: Y has two predecessors and in this case the reasoning is the same as in **3**, or X can have two predecessors and in this case we can perform an intermediate merging since the two clusters ($pred(X)$) will have f values greater than the ones of theirs successors.

Knowing these, we can eliminate the recursive refinement step by adding test conditions for these cases in order to deal with this situations. In order to have a weakly indexed 2-3 hierarchy we introduce these conditions each time we create a cluster: eliminate the successors of the new cluster if their value is the same with the one of the new cluster, and if they're not the proper intersection of two clusters. In this way the 2-3 hierarchy obtained will be weakly indexed by the mapping f.

2.4 Aggregation Index

In this section we will analyze analyze the ways of defining the indexing formulas and their influence on the created 2-3 hierarchy, in order to determine the most suitable indexing formula.

One of the ways of measuring the "quality" of a ascending hierarchical clustering method is given by its induced dissimilarity matrix. This is obtained using the indexing

level f of its created clusters. The indexing of a cluster has thus a great influence on the method's quality.

In this context, we analyze the influence on the 2-3 AHC algorithm of the:

- different ways of defining a dissimilarity,
- different dissimilarity measures (*single-link*, *complete-link* and sometimes *average-link*),
- different indexing formulas by an isotone mapping f, for the created 2-3 hierarchy.

The main concern here are the clusters that properly intersect themselves, since they are a characteristic of the 2-3 hierarchies that differentiate them from the classical hierarchies.

2.4.1 Context

We first recall some used definitions. During the 2-3 AHC algorithm, two clusters can be merged if they are closest - in the sense of a chosen *link of aggregation* hereafter denoted μ and called simply *link*. In the 2-3 hierarchical case each cluster can be merged at most two times, leading thus to clusters that properly intersects. The most usual links are the *single-link*, the *complete-link*, the *average-link*, the *Ward criterion*, etc. When two clusters X and Y are merged, the link $\mu(X, Y)$ between these two clusters can be interpreted as a measurement, denoted $f(X \cup Y)$, of the degree of heterogeneity of the new cluster $X \cup Y$. At the end of the mergings when the initial cluster E is created, the created clusters structure denoted $\mathcal{S}$, can be used to extract the *induced dissimilarity*, δ' where: $\delta'(x, y) = \min\{f(X) : X \in \mathcal{S} \text{ and } x, y \in X\}$.

The map f that associates each cluster to its degree of heterogeneity, is not necessarily an index (hierarchies) or a weak index (2-3 hierarchies), in the sense defined in Section 2.1.1, so that a refinement step (removing of certain clusters) is performed after the last merging of the algorithm in the initial 2-3 AHC algorithm. The refinement starts from the last formed cluster, E, and proceeds "downwards" with the clusters eliminations until it reaches the singletons.

More precisely, the purpose of the refinement step is to eliminate any cluster which does not influence the induced dissimilarity matrix. In the hierarchical case, these clusters are the ones found on the same f level with their predecessor and the result of the refinement phase is an indexed hierarchy.

For the 2-3 hierarchies these clusters are the ones found on the same level with their predecessor(s), excepting the case when such a cluster is a proper intersection and can influence the induced dissimilarity. For example in Figure 2.29, Y is on the same f level with one of his predecessor, $X \cup Y$, but cannot be eliminated since it is the proper intersection of $X \cup Y$ and $Y \cup Z$ and its elimination would influence the induced

dissimilarity. The result of the refinement phase in this case is a weakly indexed 2-3 hierarchy.

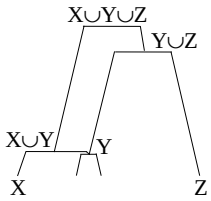


Figure 2.29: Clusters on the same level in a 2-3 hierarchy

The quality of a hierarchical classification method consists in its representation of the initial dissimilarity matrix by the created structure, i.e. the induced dissimilarity matrix (see above definition). Thus, after the refinement, the induced dissimilarity matrix is compared with the initial dissimilarity matrix using different formulas (e.g. the *Stress* formula, the *Pearson coefficient* or a simple difference) in order to determine the "quality" of the obtained structure.

In conclusion, contrary to the classical AHC, in the 2-3 AHC:

- a cluster can be merged with at most two other disjoint clusters,
- we will obtain in the refinement step a weak index of the structure, since here we have clusters that properly intersects and their elimination can influence the induced dissimilarity matrix (see Figure 2.29).

2.4.2 Link

The link defines the dissimilarity μ between two clusters based on their elements. Some well known examples of dissimilarity measures are: the *single-link*, the *complete-link*, the *average-link* etc. These dissimilarities are well defined for the hierarchies, but we need to analyze them in the context of the 2-3 hierarchies, i.e. in the context of clusters that *properly intersect* themselves such as X and Y from Figure 2.31: $X \text{p}\cap Y \Rightarrow X \cap Y \notin \{\emptyset, X, Y\}$. For this we will first take a look at the ways of defining a dissimilarity between clusters in Section 2.4.2.1.

Since the *proper intersection* of two clusters differentiates a hierarchy from a 2-3 hierarchy, we will also analyze its influence on the 2-3 AHC algorithm (see Section 2.4.2.2).

2.4.2.1 Dissimilarity definitions

We have three options when computing the dissimilarity between two clusters that properly intersect themselves:

(1) to exclude the common part of the clusters when we compute the dissimilarity. In Figure 2.31 we have: $\mu(X,Y) = \mu(X-Y, Y-X) = \mu(X',Y')$. The general formulation would be:

$$\mu_1(X,Y) = formula\{d(x,y) : x \in X-Y,\ y \in Y-X\},$$

where $formula$ can be min, max, $average$ etc;

(2) to take into account also the common part when we compute the dissimilarity. The general formulation would be:

$$\mu_2(X,Y) = formula\{d(x,y) : x \in X,\ y \in Y\};$$

(3) to consider all the distances inside the formed cluster, $X \cup Y$, when computing the dissimilarity. This would resume to:

$$\mu_3(X,Y) = formula\{d(x,y) : x \in X \cup Y,\ y \in X \cup Y\}.$$

2.4.2.2 Dissimilarity definitions and 2-3 AHC Algorithm

We have seen that using the *extended indexing formula** we will avoid any risk of level inversions in the created 2-3 hierarchy, but this formula can influence the execution of the 2-3 AHC algorithm. We analyze here the use of the three previous dissimilarity definitions during a proper intersection in the (general) 2-3 AHC algorithm defined in [Ber02d] and in particular when these are applied for the *single-link*, the *complete-link* and the *average-link*.

We know [Ber02d] that the 2-3 AHC algorithm merges at each step the pair of candidates clusters found at the minimum dissimilarity, just like the AHC algorithm. Lets suppose that we merge two clusters Z and Y' and that one of them, say Z has another predecessor, X (Figure 2.31). The new formed cluster Y, will have a maximal degree of heterogeneity among the other candidate clusters, due to the merging of two disjoint clusters, Z and Y', at the maximal dissimilarity so far. As in the AHC case, the dissimilarity between the resulting cluster and all other disjoint candidate clusters will be higher or equal to $\mu(Z,Y')$. But the dissimilarity value between the two clusters that properly intersect themselves, X and Y, depends on the chosen dissimilarity definition from above (1-3) and naturally on the chosen dissimilarity measure.

When performing a merge (Figure 2.31), the 2-3 AHC algorithm takes the candidate clusters found at minimum dissimilarity, and thus it is obvious that:

$$\mu(X',Z) \leq \mu(Z,Y') \leq \mu(X',Y') \tag{2.13}$$

Using this and μ_1, we will always have $\mu(X,Y) = \mu(X',Y') \geq \mu(Z,Y') \geq \mu(X',Z)$ for any dissimilarity measure.

Next, we analyze the use of μ_2 and μ_3 from above with the single-link, complete-link and average-link.

* $f(X \cup Y) = max\{f(X), f(Y), \mu(X,Y)\}$)

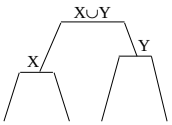


Figure 2.30: No intersection

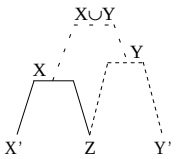


Figure 2.31: Proper intersection

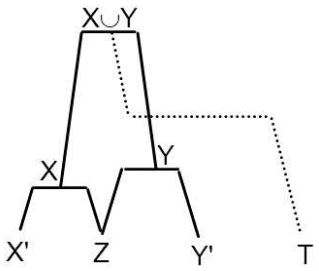


Figure 2.32: Levels inversion

For the *single-link* we cannot use μ_2 and μ_3, since the dissimilarity between X and Y will be 0 if we consider their common part, Z (Figure 2.31) and thus we will find ourself in the hierarchical case after the refinement at the end of the 2-3 AHC algorithm [Jul02a].

For the *complete-link*, when we compute $\mu(X,Y)$ with $X_{\mathrm{p}}\cap Y$ (Figure 2.31), the dissimilarity $\mu(X',Y')$ will always influence $\mu(X,Y)$, regardless of the chosen dissimilarity definition. Since it uses the maximum value, it results that $\mu(X,Y) \geq \mu(X',Y')$ and using (2.13) it follows that $\mu(X,Y) \geq \mu(X',Y') \geq \mu(Z,Y') \geq \mu(X',Z)$. All three definitions can be used with the complete-link but we will analyze in Section 2.8 their use, to select the most suitable one.

If we use the *average-link* and we also consider the common part of X and Y (definitions μ_2 and μ_3), we will have a small dissimilarity between X and Y, caused by the common cluster, Z. This is due to the fact that the distances between the elements of the common clusters are considered when computing the average. In this case, we can have $\mu(X,Y) < \mu(Z,Y')$ and we must use the *extended indexing formula* in order to avoid levels inversions.

A small example is presented in Figure 2.33 bellow. The average dissimilarity is computed using: $\mu(X,Y) = \frac{\sum d(x,y)}{n}$, where $x \in X$, $y \in Y$ and n is the number of possibles pairs in the numerator sum ($|X| \times |Y|$).

In conclusion, the first formula (μ_1) for defining the dissimilarity μ between two clusters that properly intersect themselves, seems to be the best formulation since it can be applied for all dissimilarities. If we use the μ_2 and μ_3 dissimilarity definitions, the 2-3 AHC algorithm can have unexpected/unwanted results if we use them with the

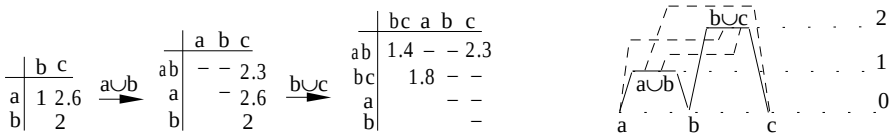


Figure 2.33: Average-link and 2-3 AHC algorithm

extended indexing formula or other formulas (see next Section 2.5), especially in the complete-link case.

2.5 2-3 Hierarchy Indexing

We study now the indexation of the (2-3) hierarchies and also the special case of the clusters that properly intersects themselves. The indexation is very important since it will influence the graphical representation (the *dendrogram*) of the (2-3) hierarchies and the refinement phase performed at the end of the algorithm. The purpose of this section is to establish which indexing formula to use for the single, complete and the average link.

As we said before, the hierarchies and the 2-3 hierarchies are indexed using an isotone mapping f, which represents the degree of heterogeneity of a cluster. This index is used at the end of the algorithm in the (optional) refinement phase that eliminates the useless clusters from the created structure. After that, this index is finally used to represent graphically the created structure (hierarchy or 2-3 hierarchy) as a dendrogram and to extract the induced distance matrix (the *ultrametric* or the 2-3 *ultrametric*). This induced distance matrix is then compared with the initial one in order to determine the structure's quality (the *Stress* coefficient, *Pearson* coefficient, etc).

The definition of this index f is influenced by the chosen dissimilarity and in the case of the 2-3 hierarchy it can be:

(**a**) $f(X \cup Y) = \mu(X, Y)$;
(**b**) $f(X \cup Y) = max\{\mu(X, Y),\ f(X),\ f(Y)\}$.

As follows we will refer to the first indexing formula (**a**) as the *normal indexing formula* and to the second one (**b**) as the *extended indexing formula*.

The *normal indexing formula* represents the dissimilarity at which the two clusters have been merged and is used in the AHC algorithm. Since the AHC algorithm merges at each step the minimal dissimilarity pair of clusters, it results that $f(X \cup Y) \geq f(X)$ and $f(X \cup Y) \geq f(Y)$, where $(X, Y) \in \mathcal{M}_i$ is the candidates pair that are merged. This give us an isotone index f of the created hierarchy. This reasoning does not applies for the 2-3 AHC algorithm. This is due to the fact that inversions between the levels of the created 2-3 hierarchy can appear using this formula (see Section 2.2.2.1). In order

to avoid this, an extension of this formulation is used for the single-link (see below the *double indexing*).

The *extended indexing formula* represents the heterogeneity degree inside the new formed cluster and is used in the 2-3 AHC algorithm in order to avoid the situations where $f(X) > f(X \cup Y)$ or $f(Y) > f(X \cup Y)$, situations also denoted here as *levels inversions*. These situations are presented in Section 2.4.2.2 and concern the average-link case (smaller dissimilarity, Figure 2.33) and the levels inversions for the single-link and the complete-link (higher dissimilarity, cf. Figure 2.32) and Section 2.2.2.1), when using the normal indexing formula.

It has been proved in [Jul02a] that the extended indexing formula can produce inversions in the resulting 2-3 ultrametric for the single-link. On the other hand, this formulation is suitable for the complete-link, since it maximizes the level of the new cluster. But if we use this formula, all the clusters resulted from the merging of $X \cup Y$ (plus any resulting cluster) and different clusters situated at a dissimilarity inferior to $\mu(X, Y)$, will possibly be eliminated during the refinement phase of the algorithm (Figure 2.34) when we use this extended formulation.

In both indexing cases, (**a**) and (**b**), $f(X \cup Y)$ is influenced by $\mu(X, Y)$ and moreover $f(X \cup Y) \geq \mu(X, Y)$.

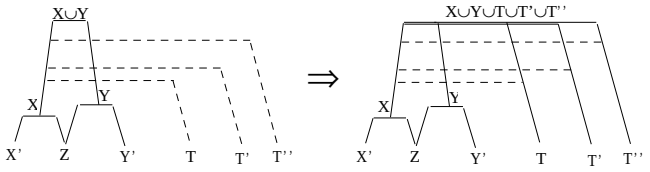


Figure 2.34: Using the *extended indexing formula*

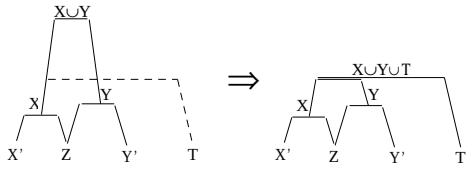


Figure 2.35: Using the *double single-link*

We present here an extension of the normal indexing formula, called the *double single-link* formula, introduced in [Jul02a] in order to avoid inversions in the resulting 2-3 ultrametric for the single-link when using the extended indexing formula.

For this, the normal indexing formula is slightly altered in order to have a pre-indexed 2-3 hierarchy, i.e. in order to avoid having $f(X \cup Y) > f(X \cup Y \cup T) = \mu(X \cup Y, T)$, cf. Figure 2.32. This new indexing formula is called the *double single-link* and is defined as follows:

- $f(X \cup Y) = \mu(X, Y)$ for disjoint clusters,
- $f(X \cup Y) = min\{\mu(X', Y'), \mu(X \cup Y, T)\}$ where T is the closest candidate cluster to $X \cup Y$, for clusters that properly intersects (Figure 2.35).

Note: Since the double single-link does not concerns the dissimilarity calculus (the link) between clusters, but only the indexing of the structure, we will simply call it the *double indexing formula*.

If we extend this *double indexing formula* to all dissimilarities, for example for the complete-link, it can have the same effect as the extended indexing formula for single-link (inversions in the resulting 2-3 ultrametric). For example in Figure 2.35: reducing the level of $X \cup Y$ will cause smaller values in the resulting 2-3 ultrametric compared with the ones in the initial dissimilarity.

An example of using the double indexing formula is presented in Figure 2.35, where $f(X \cup Y) > \mu(X \cup Y, T)$. Here, when we create $X \cup Y \cup T$, we will reduce $f(X \cup Y)$ to $\mu(X \cup Y, T)$.

In conclusion, so far the best indexing formulas for the dissimilarity measures are:

- for the single-link: the *double indexing formula*. The normal indexing formula can cause levels inversions, while the extended indexing formula can cause inversions in the resulting 2-3 ultrametric.
- for the complete link: the *extended indexing formula*. As in the single-link case, the normal indexing formula can cause levels inversions, while the double indexing formula can cause inversions in the resulting 2-3 ultrametric.
- for the average-link: the *extended indexing formula* and the double indexing formula but clusters will tend to be on the same level with their predecessor in a proper intersection. Thus will most probably lead to a poor 2-3 hierarchy (very similar to a hierarchy). The normal indexing formula can cause levels inversions.

2.6 Proper Intersection Analysis

We analyze in this section, the reasons for the level inversions (see Section 2.2.2.1), the measures to avoid them (the extended indexing formula) and their influence on the created 2-3 hierarchy.

2.6.1 Inversion problem

We begin by recalling some basic notions.

The main difference between 2-3 hierarchies and hierarchies are the clusters that properly intersect themselves: X *properly intersects* Y, if $X \cap Y \notin \{\emptyset, X, Y\}$ (Figure 2.36). In the hierarchy case, when a cluster is formed, his successors can't be merged with another cluster and they are eliminated from the candidate cluster set (in Figure 2.30, X and Y are no longer candidate clusters).

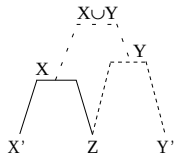


Figure 2.36: Proper intersection in a 2-3 hierarchy

In a 2-3 hierarchy, these successors are kept as candidates and they can be merged only with other maximal clusters (see Section 2.3). For example in Figure 2.36, after the merging of X' and Z the resulting cluster, X, is a candidate cluster along with his successors, X' and Z. Since these successors are smaller clusters than X, in case of same dissimilarity value with another cluster, they will be *preferred* for the next merging by the aggregation index (dissimilarity measure μ), which tries to minimize the dissimilarity between candidate clusters and other given criterion (cardinality, creation order, etc). Also generally speaking, is it more likely that a cluster Y' is "closer" to a subset (e.g. a successor) of another cluster X (Z in this case), then to X itself. Thus, for example Z can be merged with another cluster, Y', forming a new cluster, Y (see Figure 2.36).

At this stage we have two clusters that properly intersect themselves, X and Y with $X \cap Y = Z$. We have seen in Section 2.3 that these two clusters and their predecessors cannot be merged with another cluster disjoint of them and that only X and Y will necessarily be merged together in a later step of the algorithm.

Next we will analyze this situation and the outcoming of how the proper intersection was performed.

2.6.2 *Blind Merging*

The main problem is that at the merging of Z and Y', we don't consider the dissimilarity between X' and Y' which will influence the heterogeneity degree of the future cluster $X \cup Y$ and sometimes (see Section 2.7.2) the dissimilarity between $X \cup Y$ and the rest of the candidate clusters. We denote as a ***blind merging*** the indirect merging of X' and Y' through the future merging of X and Y.

For example, if X' and Y' are *opposed* (in the sense of μ), the dissimilarity between them will have a high value, resulting in a high degree of heterogeneity of the future

cluster $X \cup Y$ for some dissimilarity measures (single and complete link). Moreover, the dissimilarity between X and Y will have a high degree (value) causing the "block" of X and Y's merging, until the minimum dissimilarity in the initial 2-3 AHC algorithm execution reaches $\mu(X, Y)$ (see Section 2.7.2).

We call this merging *blind merging* of X' and Y' may cause unwanted effects, such as *levels inversions* ($f(X \cup Y) < f(X)$ or $f(X \cup Y) < f(Y)$), if one will use the normal indexing formula (Figure 2.32). But as we saw before (cf. Section 2.5), we will use the extended indexing formula (cf. Figure 2.34) or the double indexing formula (cf. Figure 2.35). This behaviour can cause the elimination of some clusters during the refinement step, clusters that will be on the same level with their predecessor due to the use of these indexing formulas.

Another issue concerning the blind merging is the fact that the induced dissimilarity matrix will not be comparable with the one induced by the classical AHC (the ultrametric). Indeed, if clusters X' and Y' are *opposed* (in the sense of μ), then it is very likely that they will be merged together later in the classical AHC algorithm execution. But in the 2-3 AHC algorithm, they will be merged together sooner, inducing thus a dissimilarity matrix non-comparable with the classic ultrametric. A more detailed example in given in Appendix A.

The blind merging is also the cause of the 2-3 hierarchy levels inversion presented in Section 2.2.2.1 if using the normal indexing formula.

In conclusion, the influence of the dissimilarity between X' and Y' on the created cluster can be on one hand, a positive source of information compared to the classical hierarchy (a new cluster or level in the dendrogram) and on the other hand, a "forced" merging of X' and Y' which will create a high heterogeneity degree cluster.

Before we analyze the blind merging influence (see Section 2.7.2) we will first take a look at the 2-3 AHC algorithm execution in this situation (see next Section 2.7).

2.7 2-3 AHC Algorithm Execution

In this Section, we study the 2-3 AHC algorithm execution when two clusters properly intersect themselves and the influence of the blind merging on this execution.

2.7.1 Proper Intersection During Algorithm Execution

When we have two clusters X and Y that properly intersect themselves (Figure 2.36), we have two options for the execution of the 2-3 AHC algorithm:

(**i**) we can leave X and Y and merge the next two closest clusters in the sense of μ; X and Y will be merged together later during the algorithm when

$\mu(X,Y) \leq \mu(S,T) : S,T \in \mathcal{M}_i - \{X,Y\}$ (they can also be the next merged clusters). This is the "normal" execution of the algorithm [Ber02b].

(**ii**) we can "force" the merging of X and Y since we know that they will be merged only together at a later step of the algorithm (see Chapter 3). This we call it an *intermediate merging* between X and Y and is influencing the ulterior algorithm execution.

In the first case (**i**), during the 2-3 AHC algorithm, the clusters merged at each step will be the candidate clusters found at minimal dissimilarity. But we can still have inversions in the levels of the formed clusters (Figure 2.32 and Section 2.2.2.1) when using the normal indexing formula. Here we need to use the extended indexing formula (for the complete link), the *double indexing* formula (for the single-link) or another indexing formula that avoids these inversions. As we said before, the extended indexing formula is not a good choice for the single-link. But the use of the double indexing formula with the single-link can also create levels inversions in the resulting 2-3 hierarchy in the normal algorithm execution. This can occur in a proper intersection: a merged cluster can have a bigger f level compared with the resulting cluster (see a detailed example in Appendix B).

This situation can be avoided if we use the *normal indexing formula* along with a test (or if we use just the intermediate merging (**ii**)). We can use the normal indexing formula (**a**) since the f value of a created cluster does not influence the future mergings using the normal algorithm execution, followed by a *level test* just after the creation of a cluster.

Thus for the single-link, we always use $f(X \cup Y) = \mu(X,Y)$, but we make the following test each time we create a cluster $X \cup Y$:

"if $f(X) > f(X \cup Y)$ or/and $f(Y) > f(X \cup Y)$, then $f(X) = \mu(X,\, Y)$ or/and $f(Y) = \mu(X,\, Y)$".

For the complete link, the normal execution will "block" the clusters that properly intersects themselves until the minimal dissimilarity will be $\mu(X,Y)$, then it will merge them together.

As follows, we will analyze the second case from above, the intermediate merging case (**ii**).

We have seen in Section 2.3 that after the creation of Y (Figure 2.36), the clusters X', Z and Y' are no longer candidate clusters and X and Y can be merged only with each other. So if we merge X and Y we will have a candidate cluster $(X \cup Y)$ that can have an elevated degree of heterogeneity and implicitly of the f value due to the dissimilarity between X and Y (caused by X' and Y'). But we can use different indexing formula in order to avoid unwanted situations, such as levels inversions caused by high level clusters.

This intermediate merging will give the new cluster $X \cup Y$, the possibility of being merged with another cluster, before the moment when the minimum dissimilarity between the candidate pairs of clusters will be equal or greater than $\mu(X, Y)$ (the condition from case **i**). Assuming that $X \cup Y$ will be merged with a cluster T (see Figure 2.32), where $\mu(T, X \cup Y) < \mu(X, Y)$, it results that $\mu(T, X \cup Y) < f(X \cup Y)$.

Using the intermediate merging, the cluster $X \cup Y$ will not be "blocked" until the minimum dissimilarity for choosing the merging clusters will be $\mu(X, Y)$, and it can be merged with other candidate clusters immediately. But the problem is that a cluster with high heterogeneity degree (or indexing level) can be created and we need to use different indexing methods for each dissimilarity to avoid levels inversions in the 2-3 hierarchy (see Section 2.5). For the single-link, beside the *normal indexing* formula with the test, we can use here also the *double indexing* formula without any risk of levels inversions (Appendix B).

For the complete-link in the intermediate merging case, the extended indexing formula is the more suitable one since it maximizes the f level of the new cluster, taking into account the heterogeneity degree of its successors.

2.7.2 Blind Merging's Influence

If we consider the case of clusters that properly intersect themselves, Figure 2.36, we have seen that the blind merging can create a high heterogeneity degree cluster. For example in Figure 2.36, Z and Y' are merged without taking into account the dissimilarity between X' and Y', which will influence the heterogeneity degree of the future cluster $X \cup Y$. Different dissimilarity definitions (μ_3 for complete-link) or indexing formulas (see Sections 2.5 and 2.7) will help us avoid levels inversions in this case, but the dissimilarity between $X \cup Y$ and the other candidates can also be affected by this possibly high heterogeneity degree of $X \cup Y$.

This situation does not appear when using the single-link, since it uses the *closest element* criterion, but it can influence other links like the average and especially the maximum-link which uses the *farthest element* criterion.

In the case where the dissimilarity between the new cluster and the rest of the candidates is influenced by its high heterogeneity degree, this new cluster will become an "isolated" cluster and it will be merged only later with other candidates, causing thus information losses. A small example is presented in the Appendix A.

We must remind that this loss can be only local as in the given example, but on larger datasets it can influence the way that the rest of the clusters will be merged. This translates into a different 2-3 hierarchy, which can be richer or poorer (when comparing the induced dissimilarity) compared with the one created by the initial 2-3 AHC algorithm. This is one of the reasons for the high standard deviation obtained during experimentations in the next Chapter (see also Appendix C), along with the

fact that the behaviour of a classification method strongly depends on the used dataset [ST05].

The blind merging can be avoided by minimizing the level of the cluster that is to be formed after the intermediate merging step. This can be done by storing a second aggregation value for each pair of clusters that can be β merged. This extra value is actually the link between the two properly intersecting clusters that could result from this β merging (see also Section 3.4.1 for an example).

2.8 Complete Link Definitions

We have seen that for the complete-link we can use all three dissimilarity definitions presented in Section 2.4.2.2 in the both algorithm executions (normal and with intermediate merging). The only indexing formula to use in this case is the *extended indexing formula*:

- $f(X \cup Y) = \max\{\mu(X,Y),\ f(X),\ f(Y)\}$.

Next we prove that the results of the three definitions are identical when using the extended indexing formula. The purpose is to show that μ_1 is the most suitable one, since it has a smaller time computing complexity then μ_2 and μ_3.

We will thus show that using the first dissimilarity definition and this indexing formula, will create clusters for which the level f represents its heterogeneity degree, e.g. $f(X) = \max\{d(x,y) : x \in X,\ y \in X\}$. This heterogeneity degree is actually assured by the third dissimilarity definition, μ_3.

Proposition 2.8.1 *If we use the complete-link defined as* $\mu(X,Y) = \max\{d(x,y) : x \in X-Y,\ y \in Y-X\}$ *and the extended indexing formula* $f(X \cup Y) = \max\{\mu(X,Y),\ f(X),\ f(Y)\}$, *then the* f *level of a cluster will represent its heterogeneity degree, e.g.* $f(X) = \max\{d(x,y) : x \in X,\ y \in X\}$

Proof. This can be proved by induction on the 2-3 AHC algorithm execution step, i. For $i = 0$, $f(\{x\})$ with $x \in E$ is set to 0 for all the singletons, verifying thus our hypothesis. We assume now that $f(X)$ represents the heterogeneity degree of X as defined above for all the candidate clusters X at the step $i = k$.

We analyze the heterogeneity degree of the cluster formed at the next step $i = k+1$ of the 2-3 AHC algorithm. At this step we will merge two clusters X and Y, which can be disjoint or can properly intersects themselves. All clusters created so far have their level f equal to their heterogeneity degree, defined as above.

The heterogeneity degree of the new resulting cluster $X \cup Y$ is given by:

$\max\{d(x,y) : x \in X \cup Y,\ y \in X \cup Y\} =$

$= \max\{\max\{d(x,y) : x \in X,\ y \in X\},\ \max\{d(x,y) : x \in Y,\ y \in Y\},$

$$\max\{d(x,y) : x \in X - Y,\ y \in Y - X\}\} =$$
$$= \max\{\, f(X),\ f(Y),\ \mu_1(X,Y)\} =$$
$$= f(X \cup Y).\ \square$$

In this way, the use of any dissimilarity definitions μ_1, μ_2 or μ_3 will create the same f level clusters, but their computation is different in time complexity. We will thus use for the complete-link the first dissimilarity definition μ_1, which is less time consuming.

2.9 Discussion and Perspectives

In this Chapter we have studied the 2-3 hierarchy concept proposed by P. Bertrand [Ber02d] in 2002. The 2-3 hierarchies generalize the classical hierarchies by allowing two clusters to properly intersect themselves. The created structure is richer and the initial proposed 2-3 AHC algorithm has a $\mathcal{O}(n^3)$ time complexity.

We have revealed four new properties of the 2-3 hierarchies, which will be used in the next Chapter to propose a new 2-3 AHC algorithm. These properties allowed us to propose a new merging step for 2-3 AHC algorithm (see Section 3.2.1). Also, using these properties we will specify exactly the clusters to be eliminated from the candidate set in the `Update` step and the dissimilarities to compute.

We saw that the recursive refinement step from the end of the 2-3 AHC algorithm can be avoided by performing an "on-the-fly" refinement. We called it the *integrated refinement* and we proved that it produces a weakly indexed 2-3 hierarchy. But depending on the desired analysis, one can chose not to perform the integrated or the recursive refinement which becomes thus optional.

Since the proper intersection is the main characteristic of the 2-3 hierarchies, we studied its influence on the aggregation index, algorithm execution, etc.
As concerning the aggregation indexes, for the ***single-link*** we must/can use the followings definitions or algorithm executions:

- we must use the first dissimilarity definition (μ_1) to construct 2-3 hierarchies,
- both algorithm executions can be used, but with different results:
  - *intermediate merging* (**i**): in order to avoid levels inversions and inversions in the resulting 2-3 ultrametric, we must use one of the two following indexing formula, which are equivalent in this case:
    - the *double indexing formula*,
    - the *normal indexing formula* with the level test.
  - *normal execution* (**ii**): we must use the *normal indexing formula* with the level test, but we will have information loses (poorer structure after the refinement).

As for the ***complete-link***:

- we can use the *extended indexing formula* and the first dissimilarity definition, μ_1,
- we could also use:
  - the *normal indexing formula* with the third dissimilarity definition, μ_3, or
  - the *extended indexing formula* with one of the last two dissimilarity definitions, μ_2 or μ_3,

  but with a bigger time computing complexity and same results.

The study of the proper intersection and of the 2-3 AHC algorithm execution has revealed a particular case on merging that we called the *blind merging*. Concerning its influence (Section 2.7.2), a possible way to avoid it, is to take into account the dissimilarity between the disjoint clusters in a proper intersection, before the β merging.

We saw that if we avoid the blind merging, one can build a different 2-3 hierarchy. Same thing happens if one chose to perform or not the integrated refinement. Unfortunately, we can not state that one choice is better than the other, since a choice like this at any level (integrated refinement, blind merging) will influence the way that the other remaining candidate clusters will be merged in the next steps of the algorithm (see also Appendixes A and B).

It will thus become necessarily to refer to the initial dissimilarity matrix when we need to compare the structures created by these different choices. In the next Chapter we will then use different coefficients like *Stress* or *Pearson* with regard to the initial dissimilarity when performing dissimilarities quality analysis.

Chapter 3

A New Agglomerative 2-3 Hierarchical Classification Algorithm

In this Chapter we will use the new theoretical properties of the Agglomerative 2-3 Hierarchical Classification discovered in the previous Chapter 2, to propose a new 2-3 AHC algorithm with a reduced complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 \log n)$.

We start by presenting the initial 2-3 AHC algorithm proposed in [Ber02d] (detailed version also presented). Then a short analysis of this algorithm will be provided.

Next, based on two theoretical properties from the previous Chapter, we will include an *intermediate merging* step into our 2-3 AHC algorithm. This will eliminate the blocked clusters case and will reduce the execution complexity of the merging and updating phase of our 2-3 AHC algorithm. As a second consequence, the created 2-3 hierarchy will be different from the one produced by the initial 2-3 AHC algorithm.

A different 2-3 hierarchy can be also obtained if an optional step, the *integrated refinement*, is performed. The integration of this step into the merging step, will remove the need of a recursive refinement step, present in the initial algorithm.

Finally, to avoid a special case of merging, denoted *blind merging* (see Section 2.6.2), a second 2-3 AHC algorithm variant is proposed. This variant is based on a slightly modified aggregation index. and always creates a "richer" 2-3 hierarchy than the classical hierarchy on the same dataset.

Since these 2-3 AHC algorithms (initial, with intermediate merging, with integrated refinement, avoiding blind merging) construct different 2-3 hierarchies, we will later perform some comparative tests. The execution times of each method, along with the classical AHC, are compared for the complexity validation. For the algorithm "quality", we use the *Stress* measure [JW82] to compare the created structures on simulated and real data.

We will then conclude in Section 3.7 with discussions and future perspectives.

3.1 Initial 2-3 AHC Algorithm

We recall here the initial 2-3 AHC algorithm proposed in [Ber02d] in order to generalize the classical AHC algorithm.

Initial 2-3 AHC Algorithm:

1. **Initialization**: $i = 0$; The set of clusters and the set of candidate clusters $\mathcal{M}_i$ coincide with the set of singletons of E; $f(\{x\}) = 0$, $\forall x \in E$.
2. **Merge**: $i = i + 1$; Merge a pair $\{X_i, Y_i\}$ such that $\mu(X_i, Y_i) \leq \mu(X, Y)$, among the pairs $\{X, Y\} \subseteq \mathcal{M}_{i-1}$, which are noncomparable and satisfy α or β :
   (α) X and Y are maximal, and X (resp. Y) is the only cluster susceptible to properly intersect Y (resp. X).
   (β) One of X or Y is maximal, and the other admits a single predecessor Z. No cluster is properly intersected by X, Y or Z.
3. **Update**: $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$, from which we eliminate any cluster strictly included in at least a cluster of $\mathcal{M}_{i-1}$ and in $X_i \cup Y_i$.
   Update μ by using an extension of Lance and Williams Formula.
   Update f by using $f(X_i \cup Y_i) = \max\{f(X_i), f(X_i), \mu(X_i, Y_i)\}$.
   Update $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{X_i \cup Y_i\}$.
4. **Stop test**: repeat steps 2 et 3, until the cluster E is created.
5. **Refinement**: remove from $\mathcal{S}_i$ some clusters so that f is a weak index.

As we can see, the initial 2-3 AHC algorithm consists in three phases: an *initialization phase*, a *merging phase* and a *refinement phase*. The merging phase has an additional condition (compared to the classical AHC algorithm) for the case when two clusters properly intersect each other. It operates on two sets : $\mathcal{S}_i$ which represents the clusters created so far and $\mathcal{M}_i$, the candidates for merging at each step. At the end, $\mathcal{S}_i$ will contain the whole cluster structure.

It has been proved [Ber02d] that for any choice of μ, this algorithm converges in at most $O(n^3)$, that after each step of the algorithm, the set of created clusters (completed by E) is a 2-3 hierarchy (cf. Proposition 5.4 in [Ber02d]), and that the final structure is weakly indexed.

Before making a short analysis of this initial 2-3 AHC algorithm, we give below its more detailed version from [Ber02d]:

Initial 2-3 AHC Detailed Algorithm (detailed)

STEP 1. INITIALIZATION

$i \leftarrow 1$; (*iteration number*) ; $\mathcal{S}_0 \leftarrow \{\{x\} : x \in E\}$; $\mathcal{M}_0 \leftarrow \mathcal{S}_0$

$f(\{x\}) \leftarrow 0$ and $\mu(\{x\},\{y\}) \leftarrow \delta(x,y)$, for all x, y distinct in E

STEP 2. MERGING

While ($\mathcal{M}_i$ has more than one maximal cluster) do

$i \leftarrow i+1$

Select (X_i and Y_i) such that $\mu(X_i,Y_i) \leq \mu(X,Y)$ for all noncomparable clusters X, Y in $\mathcal{M}_{i-1}$ satisfying one of the following conditions:

(α) $pred\{X,Y\} = \emptyset$ and X (resp. Y) does not properly intersect any cluster different from Y (resp. X).

(β) One and only one of the clusters X or Y admits a predecessor, say Z, and X, Y, Z do not properly intersect any cluster;

$\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{X_i \cup Y_i\}$

$\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$

Compute $\mu(X_i \cup Y_i, Z)$ for all $Z \in \mathcal{M}_i$ which is not comparable to $X_i \cup Y_i$

$f(X_i \cup Y_i) \leftarrow \max\{f(X_i), f(Y_i), \mu(X_i,Y_i)\}$

EndWhile

$\mathcal{S} \leftarrow \mathcal{S}_i$; $k \leftarrow i$

STEP 3. REFINEMENT

While $k > 1$ do

$Z \leftarrow X_k \cup Y_k$

$m \leftarrow |suc(Z)|$

$j \leftarrow 1$

While $j \leq m$ do

Denote the j^{th} element of the list $suc(Z)$ as Z_j.

If $pred(Z_j) = \{Z\}$ and $f(Z_j) = f(Z)$ then

$\mathcal{S} \leftarrow \mathcal{S} - Z$

update($suc, pred$; $\mathcal{S}$) // Update the list suc and $pred$ // after the deletion of Z_j.

$j \leftarrow m+1$

$k \leftarrow k+1$

Else

$j \leftarrow j+1$

EndIf

EndWhile

$k \leftarrow k-1$

EndWhile

The main characteristic of this 2-3 AHC algorithm is the fact that once two clusters that properly intersects themselves are created, they will remain as candidates until their dissimilarity will become the algorithm's minimum dissimilarity. Also, choosing the two merging clusters is performed by testing the minimum dissimilarity pairs until the α and β conditions are satisfied.

3.2 Using the New Theoretical Properties

We present here a new 2-3 AHC algorithm derived from the previous one and based on the properties presented in Chapter 2. The interest of this new algorithm (cf. Section 3.2.4) is two-fold: first, its principle is more similar to the principle of the AHC algorithm and second, we will see that the introduction of the intermediate merging phase, allows to reduce the complexity of the algorithm (see Section 3.4).

3.2.1 Adding an Intermediate Merge at the End of each β Merging

The following property highlights the need of merging together, after a β merging, the two clusters that properly intersect themselves.

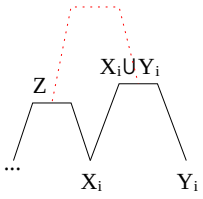


Figure 3.1: Intermediate merging

In Figure 3.1 the properly intersecting clusters are the new cluster $X_i \cup Y_i$ and the single predecessor Z of X_i (same reasoning applies if Y_i had a second predecessor).

Corollary 3.2.1 *If the merging of the i^{th} step of the algorithm is of type β, then the cluster $X_i \cup Y_i$ formed at this stage, will necessarily be merged with the predecessor of X_i or Y_i, in a later step of the algorithm.*

Proof: Let us suppose - without any loss of generality - that Z is the (only) predecessor of X_i, before the β merging of X_i and Y_i. Let us place at the end of the β merging. Clearly $X_i \cup Y_i$ is maximal and $X_i \cup Y_i \in \mathcal{M}_i$.

Suppose that Z is not maximal, then $X_i \subset Z \subset Z'$, which implies that X_i has been eliminated from $\mathcal{M}_{i'}$ $(i' < i)$ no later than during the update following the creation

of Z': this contradicts $X_i \in \mathcal{M}_{i-1}$. Thus Z is maximal, and so $Z \in \mathcal{M}_i$, because a maximal cluster cannot be eliminated from any $\mathcal{M}_j$ ($j \leq i$). It results that the clusters $X_i \cup Y_i$ and Z belonging to $\mathcal{M}_i$, are maximal and properly intersect themselves. Thus they can be merged together in an α merging, and according to Property 4 in Section 2.3.3, they will be merged together in a later step of the algorithm. □

3.2.2 2-3 AHC Algorithm's New Formulation

First, we present here a new formulation of the 2-3 AHC algorithm which basically includes and integrates Property 3 and 4 (cf. Section 2.3.3). We use here the fact that two clusters that properly intersect themselves and their successors can not bee merged with other disjoint clusters. We know then that these two clusters will be merged in a future step of the 2-3 algorithm in order to respect the 2-3 hierarchy properties and to create one (cf. Corollary 3.2.1).

In this context we chose to merge any two clusters that properly intersect themselves as soon as the second one is created. If we take the example in Figure 3.2, after clusters $\{b\}$ and $\{c\}$ have been merged, we merge directly clusters $\{ab\}$ and $\{bc\}$ into $\{abc\}$.

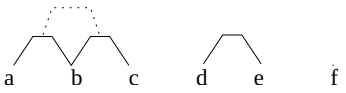


Figure 3.2: Intermediate merging

The advantage in this case is that the dissimilarities between the newly created cluster $\{bc\}$ and the rest of all candidates will not be computed. In the normal (initial) 2-3 AHC algorithm execution, these dissimilarities would have been computed or at least a test would have been performed for each candidate to determine if the dissimilarity can be computed ($\mathcal{O}(n)$ complexity). In Figure 3.2 and in the normal execution, the dissimilarities between $\{bc\}$ on one hand and $\{d\}$, $\{de\}$, $\{e\}$, $\{f\}$ on the other hand would be evaluated and possibly computed. Also clusters $\{ab\}$ and $\{bc\}$ would not be merged, but the next minimum dissimilarity pair of clusters will (for example $\{e\}$ and $\{f\}$).

The downside of adding this intermediate merging, is that we have to compute the dissimilarities between the last cluster $\{abc\}$ and the rest of the current candidates. Since the current size of $\mathcal{M}_i$ is different (possibly larger) than the size of $\mathcal{M}_i$ when merging $\{ab\}$ and $\{bc\}$ without intermediate merging, the complexity also might be different. Remark: the complexity will remain in $\mathcal{O}(n)$ since there are maximum $\lfloor \frac{3}{2}(n-1) \rfloor$ candidates, but it can be slightly bigger

The advantage is that the cluster $\{abc\}$ will be a candidate cluster in the next step of the algorithm and could be merged with other clusters to form lower heterogeneity

clusters. In the example from Figure 3.2, one can assume that there is a relatively small dissimilarity between $\{abc\}$ and say $\{d\}$ or $\{f\}$, and that these two will be merged into a new cluster creating thus a different 2-3 hierarchy.

Another advantage, is the fact that the cluster $\{ab\}$ and $\{bc\}$ and no longer "blocked" until their dissimilarity becomes the algorithm's minimum dissimilarity and the fact that we can directly eliminate $\{ab\}$ and $\{bc\}$ from the candidates set (see Section 3.2.3).

This new pseudo-algorithm formulation was also proposed in [Ber02b] *.

Preliminary Algorithm of 2-3 Hierarchical Ascending Classification

1. Initialize the set of clusters as the set of all singletons of E.
2. Merge a pair of two nearest clusters among the pairs formed by non-comparable clusters that fulfill one of the following conditions :
   (α) the two clusters X_i and Y_i are maximal.
   (β) one of the clusters X_i and Y_i is maximal and the other admits a unique predecessor which is maximal.
3. Let X_i and Y_i be the two clusters being merged at step 2. If one of these two clusters admitted a unique predecessor, say Z, at the beginning of previous step 2, then merge Z and $X_i \cup Y_i$.
4. Repeat steps 2 and 3, until the whole cluster E is merged.
5. Refinement.

As mentioned before, at the end of step 3 the sets $\mathcal{S}_i$ and $\mathcal{M}_i$ are updated and the dissimilarities between the new sets and the rest of the sets in $\mathcal{M}_i$ are computed according to the dissimilarity measure [Ber86] chosen in the beginning. In case we use the single linkage dissimilarity measure we use (2.9) to compute the dissimilarity between two sets.

So far the refinement step is identical to the initial one, but we will modify it in Section 3.2.4 based on the study from Section 2.3.4. This refinement is still a recursive one. This means that the created structure is analyzed from the last cluster formed (the E set) and for each cluster the f value is compared with the ones of its successors and in case of equality the successor is removed (it does not present any interest from the clustering point of view) excepting the case were it's a proper intersection and can influence the induced dissimilarity. Then the connections $(succ, pred)$ in the 2-3 hierarchy are "redirected" in order to maintain a 2-3 hierarchical structure. Normally after this update in the structure of the hierarchy, we should repeat the test for the new successors of the cluster if we want to make sure that we will obtain a weakly indexed structure.

*The initial 2-3 AHC algorithm didn't had the test condition for merging Z and $X_i \cup Y_i$.

3.2.3 Reformulating the Update of the Set $\mathcal{M}_i$ of Candidates

We begin with a reformulation of the update of $\mathcal{M}_i$ containing the candidate clusters (Step 3) based on Property 2 from Section 2.3.2.

Proposition 3.2.2 *In the* 2-3 *AHC algorithm, we can, without changing the results of the merging, choose $\mathcal{M}_i$ (step 3) in the following way: $\mathcal{M}_i$ equals $\mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$, from which we eliminate every successor of X_i or Y_i, and also the two clusters X_i and Y_i, if $X_i \cap Y_i \neq \emptyset$ or the merging of X_i and Y_i is of type β.*

Proof: In the initial algorithm, like in the new formulation, $\mathcal{M}_i$ is equal to $\mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$, deprived of certain clusters included in $X_i \cup Y_i$. It is thus enough to compare the two ways of defining $\mathcal{M}_i$ only for the clusters of $\mathcal{M}_{i-1}$ which are included in $X_i \cup Y_i$. We first examine the successors of X_i or of Y_i. In the initial algorithm, they don't belong to $\mathcal{M}_i$, because they are included in X_i or Y_i, and in $X_i \cup Y_i$. It is also clearly the case in the new formulation. In addition, in both ways of choosing $\mathcal{M}_i$, if a cluster W is included in one of the successors of X_i (resp. Y_i), then W does not belong to $\mathcal{M}_{i-1}$, because W was already eliminated from $\mathcal{M}_{i'}$ with $i' \leq i-1$ (we use the same arguments as for the elimination of the successors of X_i or Y_i, but to a stage previous to the formation of $X_i \cup Y_i$). Since X_i and Y_i are the only successors of $X_i \cup Y_i$, these are thus the only clusters left to examinate, in order to determine if the choice of $\mathcal{M}_i$ varies according to whether we use the initial algorithm or the new formulation.

There are only three possible cases according to whether the merging of X_i and Y_i, is:

- (a) of the type α with $X_i \cap Y_i = \emptyset$,
- (b) of the type α with $X_i \cap Y_i \neq \emptyset$
- (c) of the type β.

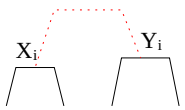


Figure 3.3: α merging of X_i and Y_i, with $X_i \cap Y_i = \emptyset$

Case (a): α merging of X_i and Y_i, with $X_i \cap Y_i = \emptyset$ (see Figure 3.3). In this case, $X_i \cup Y_i$ is the only cluster containing X_i (resp. Y_i), because X_i (resp. Y_i) was maximal before the creation of $X_i \cup Y_i$. Thus neither X_i nor Y_i are removed from $\mathcal{M}_i$ in the initial algorithm, and also in the new formulation. It results that the two formulations are equivalent here.

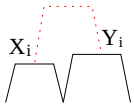


Figure 3.4: α merging of X_i and Y_i, with $X_i \cap Y_i \neq \emptyset$

Case (b): α merging of X_i and Y_i, with $X_i \cap Y_i \neq \emptyset$. Using the same argument as in case (a), we deduce that neither X_i nor Y_i are removed from $\mathcal{M}_i$ in the initial algorithm. On the other hand, X_i and Y_i do not belong to $\mathcal{M}_i$, if the new formulation is used. Using the *intermediate merging* (see Section 3.2.1) we will make sure the two clusters will be merged.

However according to Properties 3 and 4 (cf. Section 2.3.3) and Corollary 3.2.1, in the initial algorithm neither X_i nor Y_i will be aggregate during a later merging of this algorithm with another cluster. Indeed on the one hand, none of the clusters X_i and Y_i can be used for a β type merging, because X_i and Y_i properly intersect each other. On the other hand, none of the clusters X_i and Y_i can be used for an α merging, because X_i and Y_i are not maximal any more. Thus, the pairs of clusters that can be merged are the same in the two approaches.

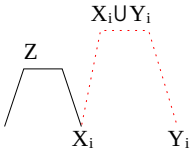


Figure 3.5: β merging of X_i and Y_i

Case (c): β merging of X_i and Y_i. Let us suppose - without any loss of generality - that Z is the (only) predecessor of X_i. Thus $X_i \notin \mathcal{M}_i$ in the initial algorithm, but $Y_i \in \mathcal{M}_i$ because Y_i is included in only one cluster ($X_i \cup Y_i$). On the other hand, X_i and Y_i do not belong to $\mathcal{M}_i$, if the new formulation is used. However according to the initial algorithm, Y_i will not be aggregate during a later merging of the algorithm (see Property 3, Section 2.3.3). Indeed, Y_i has a single predecessor $X_i \cup Y_i$ but $X_i \cup Y_i$ properly intersects Z (because Z strictly contains X_i but is disjoint of Y_i). Thus Y_i could be used neither for a β type merging, nor for an α type one. Thus, again the pairs of clusters that can be merged are the same in the two approaches, which finally proves that the new way of choosing $\mathcal{M}_i$ does not change the possibilities of merging at each iteration. □

The advantage of this approach is that the size of $\mathcal{M}_i$ is reduced and the clusters that cannot be merged with other clusters are no longer kept as candidates.

Therefore, the cluster pair satisfying the minimum dissimilarity condition will be the merging pair and no further tests are necessary to check if the pair satisfies the α and β conditions.

3.2.4 Facts

In this section we will analyze the new 2-3 hierarchies properties implications.

Fact 3.2.3 If at the end of any β merging of X_i and Y_i (i unspecified), we decide, following the Corollary 3.2.1, to merge $X_i \cup Y_i$ with the predecessor Z (of X_i or Y_i), then at the end of the so modified step 2, no cluster properly intersects a maximal cluster. In other words, *at the end of each modified step 2, the maximal clusters form a partition of* E, which underlines a strong analogy with the AHC algorithm characterized by this property.

Fact 3.2.4 For each i, the set $\mathcal{M}_i$ represents all the maximal clusters plus their successors when these successors are disjoint. This is a direct consequence of Proposition 3.2.2 and of the fact that each merging creates a maximal cluster. It results (taking into account the significant remark according to which the maximal clusters are disjoint) that one reformulate the (α) and (β) conditions in the following way, where $X, Y \in \mathcal{M}_{i-1}$: ($\alpha$) "$X$ and Y are maximal", (β) "only one of the clusters X and Y is maximal".

This can be then reduced to a single merging condition (see also Section 2.3.3) and used in the 2-3 AHC algorithm when merging two candidate clusters:

At most one is successor of another cluster from $\mathcal{M}_i$.

3.2.5 Integration of the Refinement Step into the Merging Step

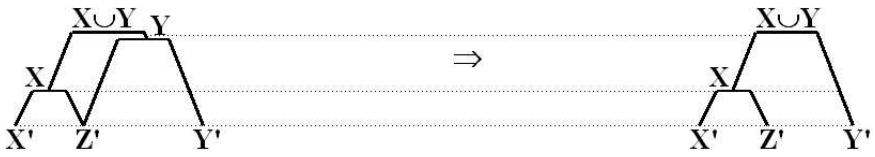


Figure 3.6: Refinement example (Fact 3.2.5)

We saw in Section 2.3.4 that one can chose to perform or not the refinement step depending on the desired analysis. The following statement considers the case when we choose to perform the refinement.

Not performing the refinement step will constructs a mapping f *isotone* in a large sense: we can have comparable clusters with the same value for f. As mentioned in Section 2.1.1 and in [Ber02c], it is more suitable to obtain a weak index f for the

created 2-3 hierarchy, especially when we are interested in comparing the resulting dissimilarities.

So we can integrate the refinement step into the merging step of the 2-3 AHC algorithm (see Section 2.3.4).

Fact 3.2.5 The refinement step can be integrated into the merging step, in order to obtain a weak indexing f. For this, each time we create a cluster $X \cup Y$, we compare $f(X \cup Y)$ with $f(X)$ and $f(Y)$. If $f(X \cup Y) = f(X)$ (resp. $f(X \cup Y) = f(Y)$), we remove X (resp. Y), provided that $X \cup Y$ is the only predecessor of X (resp. Y). This last case is illustrated in the example from Figure 3.6 where $f(X) < f(Y) = f(X \cup Y)$: Y must then be eliminated from the structure.

3.3 Proposition of a New 2-3 AHC Algorithm

Based on the previous 2-3 hierarchies properties and facts, we can reformulate the 2-3 AHC algorithm into a pseudo-algorithm as follows:

Agglomerative 2-3 Hierarchical Classification Pseudo-Algorithm

1. Initialize the set of clusters as the set of all singletons of E.
2. Merge the two nearest clusters (X_i, Y_i) among the pairs formed by non-comparable clusters with $pred(\{X_i, Y_i\}) \leq 1$.
   If $pred(\{X_i, Y_i\}) = 2$ then merge the two predecessors.
   Update $\mathcal{M}_i, \mathcal{S}_i$ and compute f, μ for the last cluster.
3. Repeat step 2 until the whole cluster E is merged.

Compared to the previous preliminary 2-3 AHC algorithm, in this version we have just replaced the α and β conditions. Next we present a more detailed version of this algorithm which integrates also the optional refinement step*.

New 2-3 AHC algorithm:

1. **Initialization**: The candidate clusters set, $\mathcal{M}_0$, is the set of singletons of E; $f(\{x\}) = 0$, $\forall x \in E$. Let $i = 0$.
2. a) **Merge**: Let $i = i + 1$; Merge two clusters X_i and Y_i which are closest (in the sense of μ) among the pairs from $\mathcal{M}_{i-1}$, which are noncomparable and such that at least one of them is maximal;

*The blind merging avoidance can be integrated on the dissimilarity calculus in the Update of μ step

b) **Intermediate Merge**: If Z is a predecessor of the cluster X_i or Y_i such that $Z \neq X_i \cup Y_i$, then merge Z and $X_i \cup Y_i$, and eliminate from $\mathcal{M}_i$ these two clusters and their successors.

3. **Refinement**: Eliminate any cluster $W \in \{X_i, Y_i, X_i \cup Y_i, Z\}$ such that W has one predecessor, W', with $f(W) = f(W')$.
4. **Update**: Update $\mathcal{M}_i$ by adding the last formed cluster and eliminating the successors of the merged clusters and also the merged clusters if they properly intersect each other.
   Update μ and f.
5. **Ending test**: Repeat steps 2-4 until E is a cluster.

Concerning this new algorithm, we may notice that facts 3.2.3 and 3.2.4 imply that the clusters generated by the new merging step 2, form a 2-3 hierarchy. The integration of the refinement step inside the loop defined by steps 2-5, ensures that the clustering structure is weakly indexed by f, whereas it is clear that the deletion of some clusters having only one predecessor, does not change the property for the generated clusters to form a 2-3 hierarchy.

A more detailed version of this new 2-3 AHC algorithm is given bellow:

New 2-3 AHC Algorithm (detailed)

STEP 1. INITIALIZATION

1. $i \leftarrow 0$; the iteration number
2. *while* $x \in E$
3. $\quad \mathcal{S}_0 \leftarrow \mathcal{S}_0 \cup \{x\}$
4. $\quad f(\{x\}) \leftarrow 0$
5. $\quad \mu(\{x\}, \{y\}) \leftarrow \delta(x, y)$
6. $\quad$ *update* ordered structure
7. *end while*
8. $\mathcal{M}_0 \leftarrow \mathcal{S}_0$

STEP 2. MERGING

9. *while* $\mathcal{M}_i$ contains merging pairs
10. $\quad i \leftarrow i + 1$
11. $\quad$ *get* (X_i, Y_i) non-comparable to merge
12. $\quad$ *create* a $X_i \cup Y_i$ with successors X_i, Y_i
13. $\quad f(X_i \cup Y_i) \leftarrow \max\{f(X_i), f(Y_i), \mu(X_i, Y_i)\}$
14. $\quad \mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{X_i \cup Y_i\}$
15. $\quad \mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$

16. *if* $f(X_i \cup Y_i) = f(X_i) = f(Y_i)$ then
17. *if* $pred(X_i) = \{X_i \cup Y_i, Z\}$ ¶
18. $succ(Z) \leftarrow \{succ(Z) - \{X_i\}\} \cup \{X_i \cup Y_i\}$
19. $pred(X_i) \leftarrow pred(X_i) - \{Z\}$
20. $\mathcal{M}_i \leftarrow \mathcal{M}_i - succ(X_i \cup Y_i)$
21. *end if*
22. $\mathcal{S}_i \leftarrow \mathcal{S}_i - \{X_i, Y_i\}$
23. $\mathcal{M}_i \leftarrow \mathcal{M}_i - \{X_i, Y_i\}$
24. $succ(X_i \cup Y_i) \leftarrow succ(\{X_i, Y_i\})$
25. $pred(succ(\{X_i, Y_i\})) \leftarrow \{X_i \cup Y_i\}$
26. *else*
27. *if* $f(X_i \cup Y_i) = f(X_i)$ ¶ then
28. $\mathcal{S}_i \leftarrow \mathcal{S}_i - X_i$
29. $\mathcal{M}_i \leftarrow \mathcal{M}_i - \{X_i, succ(Y_i)\}$
30. $succ(X_i \cup Y_i) \leftarrow \{\{Y_i, succ(X_i)\} - \{X_i \cap Y_i\}\}$
31. $pred(succ(X_i) - \{X_i \cap Y_i\}) \leftarrow \{X_i \cup Y_i\}$
32. *if* $pred(X_i) = \{X_i \cup Y_i, Z\}$ then
33. *do* 22. 23. 24.
34. *else*
35. *if* $pred(Y_i) = \{X_i \cup Y_i, Z\}$
36. $(X_i, Y_i) \leftarrow (X_i \cup Y_i, Z)$
37. *restart* from 12.
38. *end if*
39. *end if*
40. *else*
41. $\mathcal{M}_i \leftarrow \mathcal{M}_i - succ(\{X_i, Y_i\})$
42. *if* $X_i \cap Y_i \neq \emptyset$ then
43. $\mathcal{M}_i \leftarrow \mathcal{M}_i - \{X_i, Y_i\}$
44. *end if*
45. *if* $pred(\{X_i, Y_i\}) = \{X_i \cup Y_i, Z\}$
46. $(X_i, Y_i) \leftarrow (X_i \cup Y_i, Z)$
47. *restart* from 12.
48. *end if*
49. *end if*
50. *end if*
51. *compute* μ(new cluster, rest of the candidates)
52. *end while*

¶We assume without any loss of generality that X_i satisfies the condition. For Y_i we have the same reasoning

The main difference between our 2-3 AHC algorithm and the initial one, is the intermediate merging step which can sometimes generate a different 2-3 hierarchy than the one produced by the initial 2-3 AHC algorithm. Also we saw in Section 2.3.4, that one can have different 2-3 hierarchies whether we choose to execute the integrated refinement or not. Moreover, we can create different 2-3 hierarchies if we choose to avoid the blind merging case (see Section 2.6.2).

Thus we have four variants of our 2-3 AHC algorithm, based on the different options that we choose:

1. 2-3 AHC algorithm with intermediate merging and no refinement;
2. 2-3 AHC algorithm with intermediate merging and integrated refinement;
3. 2-3 AHC algorithm with intermediate merging, no refinement and avoiding the blind merging;
4. 2-3 AHC algorithm with intermediate merging, integrated refinement and avoiding the blind merging.

Before comparing these 2-3 AHC algorithm variants with the classical AHC one and the initial 2-3 AHC algorithm, we first analyze our 2-3 AHC algorithm complexity.

3.4 Complexity Analysis of our 2-3 AHC Algorithm

We used a three level ordered data structure for data matrix storage. This has a direct impact especially on the INITIALIZATION step, and so we analyze next the impact of this data structure on the 2-3 AHC algorithm execution.

3.4.1 Data Matrix Indexing

In order to store and manage the matrix containing the link values between clusters, which is the most time expensive operation, we propose to use an *ordered tree structure* that puts in correspondence these values and the pairs of candidate clusters. The purpose is to search among all candidate cluster pairs for merging, the one that minimize several criterions.

We use three criterions in order to choose the merging pair:

(1) *Minimal link*, since we search two closest clusters,

(2) *Minimal cardinality*, meaning the number of elements of the clusters to be merged, when we have multiple pairs at a minimal link

(3) *Minimal lexicographical order* (or *creation order*) on the clusters identifiers, when the two first criterons are satisfied by several pairs. This is actually equivalent to the creation order of the clusters.

Therefore, we have on the first level of the structure the ordered link values, on the second the ordered cardinalities of the pairs situated at the same link between clusters and on the third the lexicographically ordered identifiers. A small example is presented in Figure 3.7.

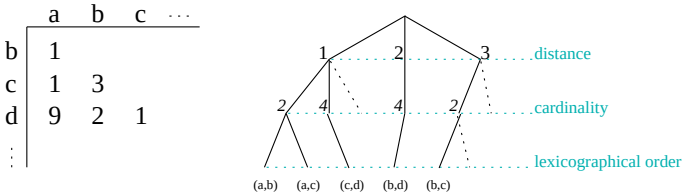


Figure 3.7: Data structure example

We know that the maximal number of non-singletons clusters in a 2-3 hierarchy is $\lfloor \frac{3}{2}(n-1) \rfloor$, where n is the number of singletons. We thus have at most $\lfloor \frac{3}{2}(n-1) \rfloor + n = \lfloor \frac{5n-3}{2} \rfloor$ candidate clusters during the 2-3 AHC algorithm execution. Since the matrix size is limited by $\lfloor \frac{n(n-1)}{2} \rfloor$, it follows that parsing this structure is performed in $\mathcal{O}(n^2)$.

Next, we will analyze the complexity of the operations on this data structure.

To access/create a cluster pair (a leaf in the data structure), we need three informations: the value of the link $\mu(X, Y)$, the cardinality of $|X \cup Y|$, and the lexicographically ordered clusters identifiers. These can be each computed in $\mathcal{O}(1)$. Having these we can access in $\mathcal{O}(\log n)$ the needed value on each level.

In the same way, the deletion of a cluster pair can be done in $\mathcal{O}(\log n)$. But during the algorithm execution, we need to eliminate all the pairs containing at least a non-candidate cluster (five at most at each step of the algorithm). The purpose is to have a structure containing only candidate pairs of clusters, and just pick the first one in the structure without performing any other tests. Thus during the 2-3 AHC algorithm execution, after each step last's merging (intermediate or normal), we must eliminate at most $5\lfloor \frac{5n-3}{2} \rfloor - 1$ pairs. Since the access to a pair is in $\mathcal{O}(\log n)$, it follows that all the pairs containing a non-candidate cluster are deleted in $\mathcal{O}(n \log n)$.

Knowing that we have at most $\lfloor \frac{3}{2}(n-1) \rfloor$ mergings, and that at each merging one must eliminate at most $5(\lfloor \frac{5n-3}{2} \rfloor - 1)$ and add at most $\lfloor \frac{5n-3}{2} \rfloor$ pairs, we have a global complexity of $\mathcal{O}(n^2 \log n)$ in manipulating this structure. This will thus not impact on the global algorithm execution and although it will increase the INITIALIZATION phase complexity, it will help reduce the complexity of some of the steps in the MERGING phase.

However, in the case of a β merging, one should analyze the behaviour of the above mentioned criterions since we know that the maximal cluster and the predecessor of the non-maximal cluster will be merged in the intermediate merging. In the case of the dissimilarity, we called that the "blind merging" influence, but this can also be extended to the cardinality and creation order.

For example, when we perform an β type merging ($|pred(\{X, Y\})| = 2$), the creation order criterion will not take into account the predecessor of the non-maximal cluster. Since this predecessor will be then merged with the other maximal cluster, the creation order criterion will allow later created clusters to be merged. An example on a small 2D points dataset (cf. Table 3.1 and Figure 3.8) using the complete-link is presented in Figures 3.9, 3.11 and 3.10.

	0	1	2	3	4
1	0.1294863923931817				
2	0.25567883955228315	0.12664556131187024			
3	0.37990591493620707	0.2518197754848242	0.12566370614359101		
4	0.5036240973594744	0.3769911184307748	0.2518197754848242	0.12664556131187024	
5	0.6283185307179586	0.5036240973594744	0.37990591493620707	0.25567883955228315	0.12948639239318166

Table 3.1: Example of lexicographical criteria influence

Figure 3.8: 2D Points

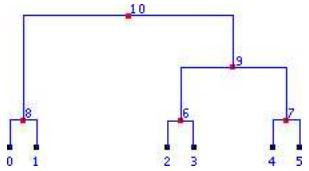


Figure 3.9: Classical AHC

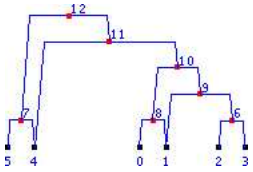


Figure 3.10: 2-3 AHC V3 avoiding BM diss

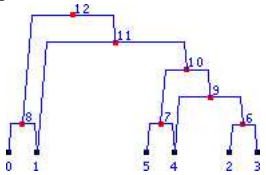


Figure 3.11: 2-3 AHC V3 avoiding BM all

The following algorithms are executed on this dataset: the classical AHC (Figure 3.9) and two 2-3 AHC algorithms avoiding the blind merging (Figures 3.10 and 3.11). The difference between the two 2-3 AHC algorithms is that the first one avoided the blind merging at the first level of the structure, the dissimilarity level, whilst the second avoided the blind merging at all three levels of the structure.

To better understand this we remind that in order to avoid the blind merging, a second value of dissimilarity is stored in the structure beside the normal one. This value is stored only for the pairs of clusters that can be β merged and is used to minimize the level of the cluster created after the α merging. We say "α merging" since if the pair chosen to be merged is a β pair, then the intermediate merging is executed (α merging). Thus, at the end of each cycle of the algorithm execution, we make sure that the last formed cluster has a minimum heterogeneity degree.

Basically, this second value is used to ensure that the last merged clusters at the end of an algorithm cycle minimize the dissimilarity criteria.

Using same reasoning, we can use two more additional values for the cardinality and creation order to ensure that these two criteria are minimized at the end of each algorithm cycle.

For example, after three cycles of the algorithms, the clusters 6, 7 and 8 are created (see Figures 3.9, 3.11 and 3.10). Here, for the 2-3 AHC methods, the candidate pairs of clusters are: $(1, 6)$, $(4, 6)$ with the same cardinality. For the 2-3 AHC avoiding the BM at dissimilarity level only (cf. Figure 3.10) we merge the pair $(1, 6)$, which in turn will involve the merging of 6 and 8 in the immediate intermediate merging step.

But, if we chose to merge the pair $(4, 6)$, the intermediate merging will group the clusters 6 and 7 which minimize all the criteria at the end of the algorithm's cycle (including the creation order). This is the case for the 2-3 AHC avoiding the blind merging at all three levels, presented in Figure 3.11.

In the case of the first 2-3 AHC algorithm, the dissimilarity matrix induced by the created 2-3 hierarchy will present a negative information gain (see Section 3.5) than the one induced by the classical hierarchy (cf. Figure 3.9), since $\{2, 3\}$ is merged with $\{0, 1\}$ instead of $\{4, 5\}$. We must specify that this is a particular case, and that one might also have a gain in the induced matrix on other datasets.

Using the 2-3 AHC avoiding the blind merging on all three levels can thus ensure that the merged pairs will minimize the three criterias and that the obtained structure will have a positive information gain compared to the classical AHC.

3.4.2 Theoretical Complexity Calculus

In this section we analyze the theoretical complexity of our new 2-3 AHC algorithm.

The complexity of the **Initialization** (step `1`) is larger than the one in the initial 2-3 AHC algorithm: $\mathcal{O}(n^2 \log n)$ instead of $\mathcal{O}(n^2)$, but the basic operations on the ordered tree structure in the following steps will be reduced to $\mathcal{O}(\log n)$ instead of $\mathcal{O}(n^2)$.

As follows we will analyze the complexity of the steps `2-4`, which are repeated until the cluster E is created, that's at most $\lfloor \frac{3}{2}(n-1) \rfloor$ times. In the **Merging** step (Step `2.a`), we first retrieve the pair that minimise our criterons, in $\mathcal{O}(1)$, and we create the new cluster $X_i \cup Y_i$ also in $\mathcal{O}(1)$. If one of the merged clusters has another predecessor, we perform an **Intermediate merge** (Step `2.b`) with the same complexity as the one before. Thus the whole complexity of the step `2` is $\mathcal{O}(n)$.

In the optional **Refinement** step (Step `3`), we will eliminate from the structure the clusters found on the same level with their predecessors when their elimination is not influencing the induced dissimilarity matrix. We update the *predecessor*, *successor* links between the remaining clusters, which is done in $\mathcal{O}(n)$, since a cluster can have at most $\lfloor \frac{5}{2}(n-1) \rfloor$ successors (cf. Property 1).

In the **Update** step (Step `4`) we first update $\mathcal{M}_i$ in $\mathcal{O}(n)$ since adding the new formed cluster is constant and since a cluster can have at most n successors to eliminate

from $\mathcal{M}_i$. In the μ update we eliminate from the structure the pairs containing at least a cluster to be eliminated. Since a pair is eliminated in $\mathcal{O}(\log n)$ and we have at most $\lfloor \frac{3}{2}(n-1) \rfloor$ clusters, we have here an $\mathcal{O}(n \log n)$ complexity. Then, the links between the new formed cluster and the rest of the candidates are computed, each in at most $\mathcal{O}(n)$, and inserted into the matrix, in $\mathcal{O}(\log n)$ each. When we want to avoid the blind merging, the only difference is that during the update, we will use two dissimilarities and also an $\mathcal{O}(\log n)$ update for the new cluster successors. Therefore, the complexity of step `4` is $\mathcal{O}(n \log n)$.

Thus, the total complexity is then reduced to $\mathcal{O}(n^2 \log n) + n \times \mathcal{O}(n \log n) = \mathcal{O}(n^2 \log n)$.

3.4.3 Experimental Validation

We have designed an object-oriented model of the algorithm, which was implemented in Java. Also a Hierarchical Clustering Toolbox was created to ease the results visualization (see Section 4 for more details).

To validate our 2-3 algorithm complexity, we analyzed the execution times on different datasets:

- random generated data: rectangular randomly generated data and sinusoidal randomly generated data with noise (see Section 3.5.3 for more details),
- real data: Abalone dataset (see Section 3.5.4 for more details).

The following six methods were executed:

1. the classical AHC algorithm,
2. the initial 2-3 AHC algorithm (denoted 2-3 *AHC ini* or 2-3 *AHC V1*),
3. our 2-3 AHC algorithm with intermediate merging and no refinement (denoted 2-3 *AHC V2*),
4. our 2-3 AHC algorithm with intermediate merging and integrated refinement (denoted 2-3 *AHC ref V2*),
5. our 2-3 AHC algorithm with intermediate merging, no refinement and avoiding the blind merging (denoted 2-3 *AHC V3*)
6. our 2-3 AHC algorithm with intermediate merging, integrated refinement and avoiding the blind merging (denoted 2-3 *AHC ref V3*).

We remind that since the blind merging doesn't influence the single link (cf. Section 2.7.2), there is only the AHC, our 2-3 AHC algorithm with or without integrated refinement (no V2 nor V3) and the initial 2-3 AHC algorithm.

We present here the execution times on the rectangle simulated data (see Appendix C for the maximum execution times). For each n, we made 10 iterations on different simulated/sampled data. The execution times on the other datasets are similar. Figures 3.12 and 3.14 present the average execution times for the single and complete linkage.

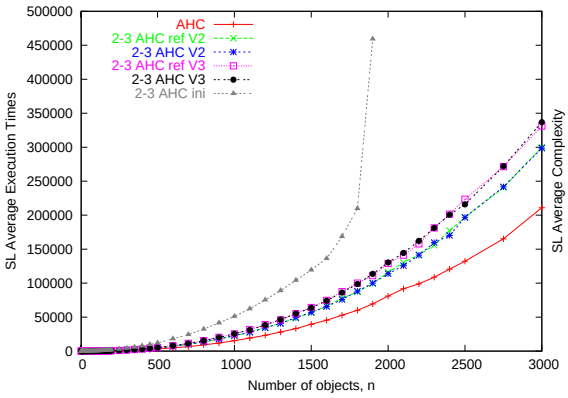


Figure 3.12: SL average execution times

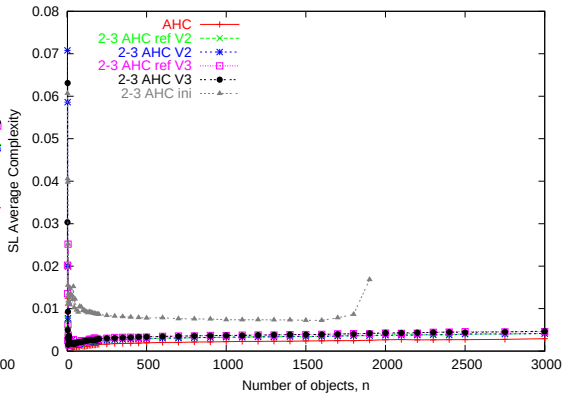


Figure 3.13: SL average complexity

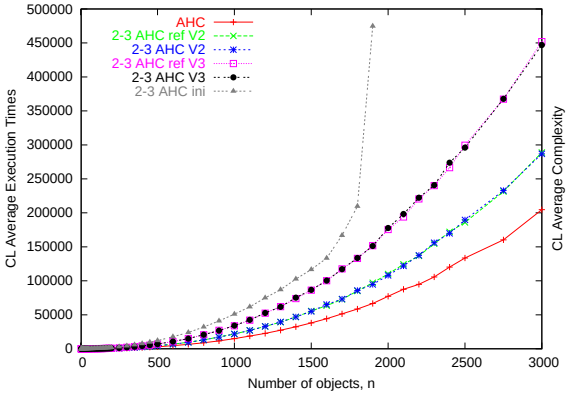


Figure 3.14: CL average execution times

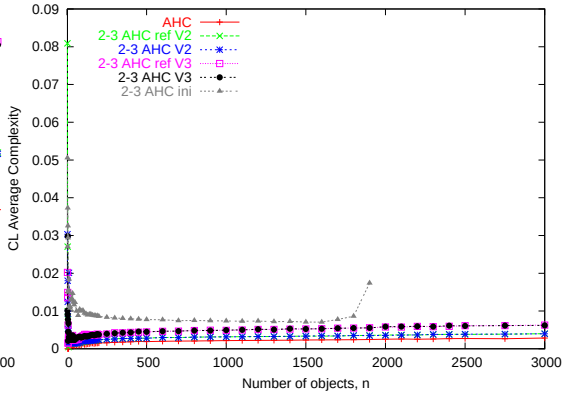


Figure 3.15: CL average complexity

To validate the algorithm complexity, we also computed the ratio $\frac{Execution\,time(n)}{\mathcal{O}(n^2 \log n)}$, where n is the number of objects. For the initial 2-3 AHC algorithm the execution times stopped at 1900 elements, while for the classical AHC and our 2-3 AHC algorithm reached 3000 elements.

As we can clearly see in Figures 3.13 and 3.15, our 2-3 AHC algorithms ratios converge to a constant proving thus that the 2-3 AHC algorithm complexity is indeed

in $\mathcal{O}(n^2 \log n)$.

3.5 Experimental Qualitative Comparison between AHC and 2-3 AHC

Having already clearly established the theoretical advantage of our 2-3 AHC algorithm (reduced computing complexity, see Section 3.4.3), in this section we will apply our 2-3 AHC algorithm variants and the initial 2-3 AHC algorithm on different datasets, and then compare the results obtained with those produced by the classical AHC algorithm. We carried out two types of comparative analysis between the classical AHC and the 2-3 AHC algorithms: on real datasets (Ruspini, Abalone) and on artificial datasets (rectangle, sinusoidal).

To compare different classification structures, there are many approaches which come especially from the hierarchies analysis field [LL95], [Pod02], [Gor99], [ST05]. These as usually used to determine the more appropriate algorithm/method to be applied for a given dataset or to determine the differences in created structures. In [ST05] the authors adopted an ordinal approach, by associating *preordenations* to the various structures:

$$\forall(\{x,y\},\{z,t\}) \in E^2 \times E^2 : \{x,y\} \leq \{z,t\} \Longleftrightarrow h(x,y) \leq h(z,t)$$

In [Ler97], a *total preorder* defined of the created partitions is used to compare the created hierarchies. The cluster creation order can also be used to compare the hierarchies [You04].

Other *dendrogram descriptor*, is the *path difference* [Phi71, WC71] which is given by the number of clusters in the path between two elements x and y in the created structure. But this is not a very suited criteria to compare the structures, as it not takes into account the levels of the dendrograms.

The most suited in our case are the *matrix comparison* coefficients (measures of discordance or agreement) for the dendrogram comparison [SS73b, LL95]. This kind of "quality" measure is also used to compare the hierarchies and the pyramids on same datasets [Bri02]. To analyze the results adequacy of the classical AHC algorithm and the 2-3 AHC algorithms, we compared the resulting structures. The problem in this case is the fact that the hierarchies and the 2-3 hierarchies can be incomparable and thus not suited for a direct comparison. Also it seems more natural to compare the structures using the initial dissimilarities.

Therefore we compared the initial dissimilarity matrix, d, with the induced dissimilarity matrix δ of each executed method. Let us recall that $\delta_{i,j}$ is the minimal value of $f(X)$ where X is a cluster containing i and j. By doing this comparison, we determined how well each structures summarize the information from the initial data.

To carry out this comparison, we used the *Stress* coefficient [JW82]:

$$\Delta = \frac{\sum_i \sum_j [(d_{i,j} - \delta_{i,j})^2]}{\sum_i \sum_j d_{i,j}^2}$$

The *Stress* measures the degree of correspondence between the induced dissimilarity matrix of the executed algorithm (AHC or 2-3 AHC) and the initial dissimilarities. Its values range from 0 to 1, where 0 corresponds to identical matrices comparison value.

Other (similar) coefficients include:

- the Weighted sum of squares:
$$\Delta = \sum_i \sum_j [(d_{i,j} - \delta_{i,j})^2]$$
- the *Goodness of fit* (*Variance accounted for*): $\Delta = 1 - \frac{\sum_i \sum_j [(d_{i,j} - \delta_{i,j})^2]}{\sum_i \sum_j [(d_{i,j} - \overline{d})^2]}$
- the *Cophenetic correlation* coefficient (*Pearson* coefficient) [SR62]:
$$\Delta = \frac{\sum_i \sum_j [(d_{i,j} - \overline{d})(\delta_{i,j} - \overline{\delta})^2]}{\sqrt{\sum_i \sum_j [(d_{i,j} - \overline{d})^2] \ \sum_i \sum_j [(\delta_{i,j} - \overline{\delta})^2]}}$$

We chosen the Stress coefficient, because is the most suitable one when comparing the dissimilarity matrices in this case.

To resume, in order to perform a qualitative analysis between the methods we used the following three criteria:

- the number of created clusters by the method,
- the *Stress* [JW82] measure, to determine how well the method represents the initial data,
- the possible improvements noticed by an expert of the domain in the data representation by studying the method's output. This is most suitable for small datasets.

For the third criteria we used a small and well-known dataset of two dimensional points, the Ruspini dataset (see Section 3.5.1), on which the visual interpretation was feasible. For this first two criteria we used artificially simulated datasets (see Section 3.5.3) and also a known large dataset, the Abalone (see Section 3.5.4), which we randomly sampled.

3.5.1 Ruspini Dataset

The well-known Ruspini dataset [Rus69] contains four clusters of two dimensional points with some intermediate points (cf Figure 3.16).

	Nb clusters	Gain nb cl	Stress	Gain Stress
AHC ref	66		0.334226	
AHC	74		0.334226	
2-3AHC ref (V2 or V3)	87	31.82%	*0.254253*	*23.93%*
2-3AHC (V2 or V3)	109	47.3%	0.256294	23.92%
2-3AHC ini	*110*	*48.65%*	0.263786	21.76%

Table 3.2: The single-link results on the Ruspini dataset

The single-link results are presented in Table 3.2 below while the complete-link results are presented in Table 3.3. The structures obtained on this dataset with the classical AHC and the 2-3 AHC algorithms, lead to the following observations.

For the single-link, we had same results with or without the blind merging avoidance (V2 and V3) which are presented in Table 3.2. The 2-3 hierarchies without integrated refinement (V2 and V3) contain more clusters than the traditional hierarchy: 110 the initial 2-3 AHC, 109 the 2-3 AHC, compared to the 74 of the classical AHC. This represents 48.65% and 47.3% more clusters (which is very close to the theoretical percentage between the maximum number of classes of the two types of structure, i.e. 50%). By analyzing the Stress coefficient, one can see that the best results are obtained by our 2-3 AHC algorithm with integrated refinement, despite the fact that there are fewer clusters.

	Nb clusters	Gain nb cl	Stress	Gain Stress
AHC [ref]	74		0.305812	
2-3AHC ref V2	105	41.89%	0.286912	6.18%
2-3AHC V2	*109*	*47.29%*	0.225438	26.28%
2-3AHC ref V3	104	40.54%	*0.198684*	*35.03%*
2-3AHC V3	107	44.59%	0.198685	35.03%
2-3AHC ini	*109*	*47.29%*	0.225438	26.28%

Table 3.3: The complete-link results on the Ruspini dataset

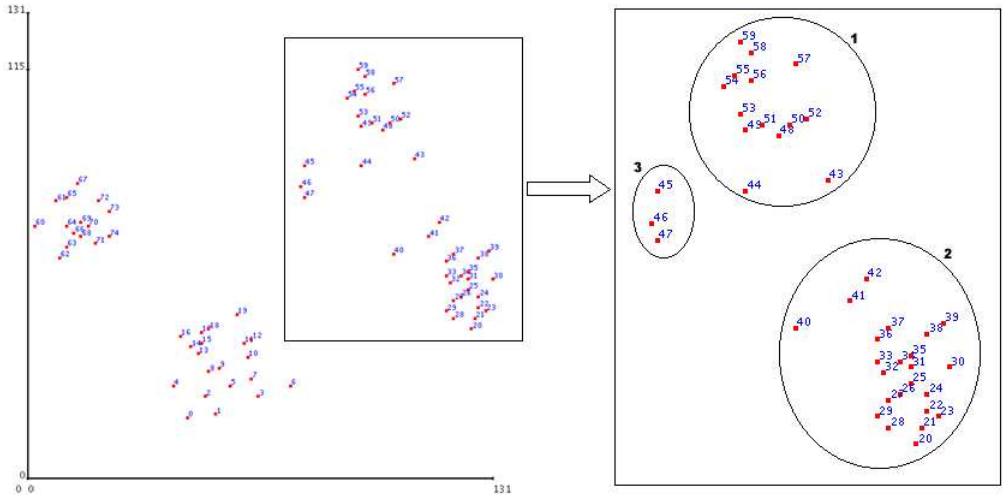


Figure 3.16: Ruspini dataset

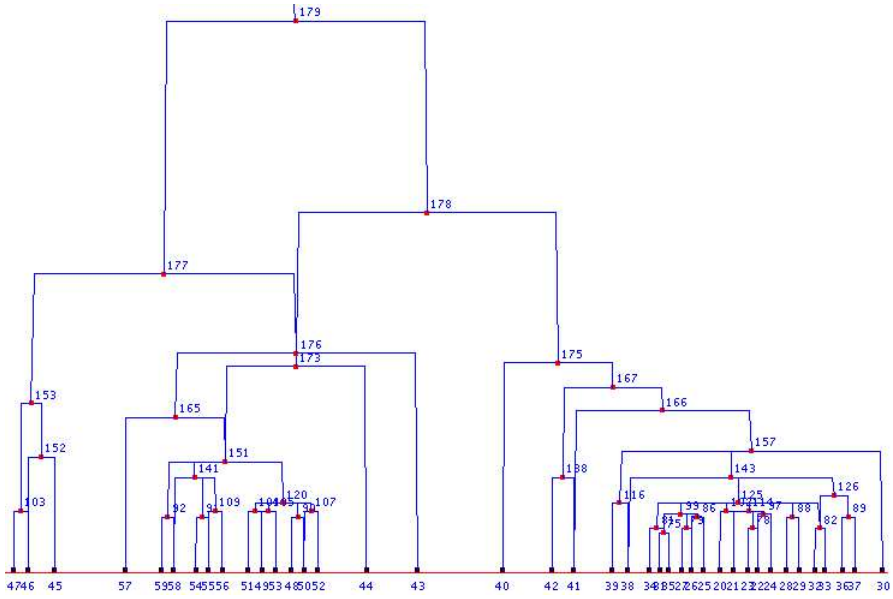


Figure 3.17: The 2-3 hierarchy on the selected points from Figure 3.16

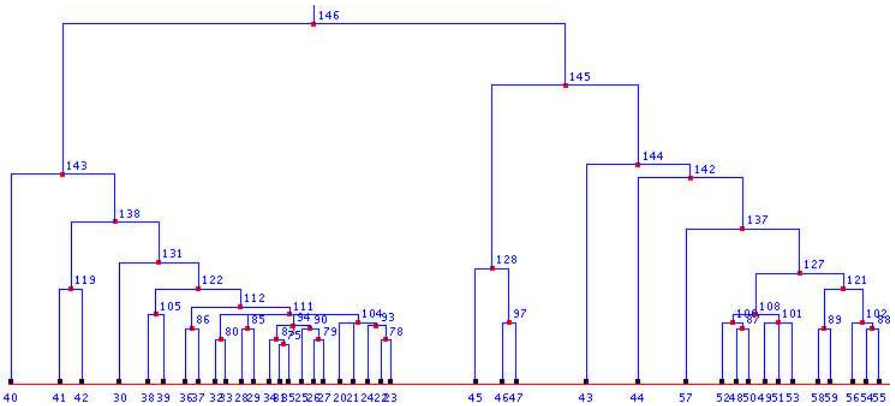


Figure 3.18: The classical hierarchy on the selected points from Figure 3.16

As concerning the complete-link results, one can see that the best results are obtained when using the 2-3 AHC algorithm that avoids the blind merging (the V3). For the AHC method, same results were obtained with or without refinement.

It results that the 2-3 AHC represents better the data structure than the classical AHC including in the case of 2-3 AHC with integrated refinement. For example, for the single-link, one can notice that for the 2-3 hierarchy in Figure 3.17), items 43, 44 of cluster 176 are closer to cluster 175 (items 20-42) then the cluster 153 (items 45, 46, 47). This is not shown by the traditional hierarchy (cf Figure 3.18).

3.5.2 Urban Itineraries

In this section we will apply our 2-3 AHC algorithm and the classical AHC algorithm on a small generated dataset of town-center urban itineraries [Bus05]. This work was realized as a part of the MobiVIP* research project of the national program Predit 3†. As a partner in the MobiVIP project, we proposed among others the use of a recommender system (*Be-TRIP* [CGT04, TCG04]) in a mobility context, to facilitate the information search and the trip's preparation and execution of a user. A more detailed description of the proposed Be-TRIP recommender system is given in Appendix F.

The purpose of the urban itineraries classification was the study of the relevance of the 2-3 AHC as a clustering method for case indexing in such a recommender system based on a Case-Based Reasoning library, CBR*Tools [Jac98] described in Section 4.3.1. The actual integration of the 2-3 AHC in the CBR*Tools is described in Section 4.3.

To analyze the relevance of the 2-3 AHC on clustering urban itineraries for the Be-TRIP recommender system, we first used the `Hierarchical Clustering Toolbox` described in Section 4.2 for results interpretation on a small generated itineraries set.

The itineraries clustering is based on the exploitation of both geographical and semantical data (road type, buildings along the road, etc.). This new use of semantic data opens new ways for itineraries recommendation, by tackling ideas as the itinerary's goal or the nature of the crossed places. Clustering itineraries has many advantages besides the possibility of choosing the most suitable one: it is also an analysis and comparison tool. This can have multiple applications: route or destination prediction, traffic anticipation, etc.

Each itinerary was split in road sections (fragments) that were analyzed for their characteristics: turning angle, length, type of present buildings, etc. Then using these informations, multiple distances between itineraries where computed based on:

- the total length and the average of maximum turning angle;

*`http://www-sop.inria.fr/mobivip/`
†`http://www.predit.prd.fr/`

- the spreading of the buildings types along the road;
- the number/proportion of common sections.

The used spreading function was proposed in [Tan05b] to compute differences between Sanskrit texts and was adapted in our case to compute the building types spreading along the analyzed itinerary. Basically, it computes the variations in the distances between the building types along the analyzed itineraries.

Then all these distances were aggregated to compute different types of distances between the itineraries. For this, we specified and developed a software called TripSimulator [Bus05] for manual itinerary generation and distances computation. Different ways of defining the distance and of comparing the road sections were tested.

Having no real data, the software was then used by George Gallais, head of the MobiVIP project, to manually generate 40 itineraries in the Antibes town center, representing four types of professional itineraries:

- Real Estate Agent (trips from 0 to 9);
- Nurse (trips from 10 to 19);
- Hotel Service (trips from 20 to 29);
- Cultural Association (trips from 30 to 39);

After computing the distance matrix on these itineraries, we applied the classical AHC along with the 2-3 AHC methods using the complete linkage (see Table 3.4 for results). The classical AHC (cf. Figure 3.19) grouped the real estate agent (cluster 71) and the hotel service (cluster 74) itineraries, while the other two groups are mixed and separated by the first group (real estate agent). Similar results were obtained using the 2-3 AHC avoiding the blind merging (2-3 AHC V3) with or without integrated refinement (cf. Figure 3.21), although the second and fourth groups (nurse and cultural association) were no longer separated, improving thus the Stress gain (cf. Table 3.4).

Using the initial 2-3 AHC, three groups were correctly constructed: real estate with cluster 88, hotel service with cluster 89 and cultural association with cluster 75. The nurse group of itineraries was divided in two clusters, 86 and 87, but they were not merged directly: 87 is merged with 90, which contains the cultural association itineraries also. Moreover, the merging strategy (hotel service close to the real estate) of the initial 2-3 AHC induced an actual loss of -22.24% in the Stress coefficient compared to the classical hierarchy (see Table 3.4).

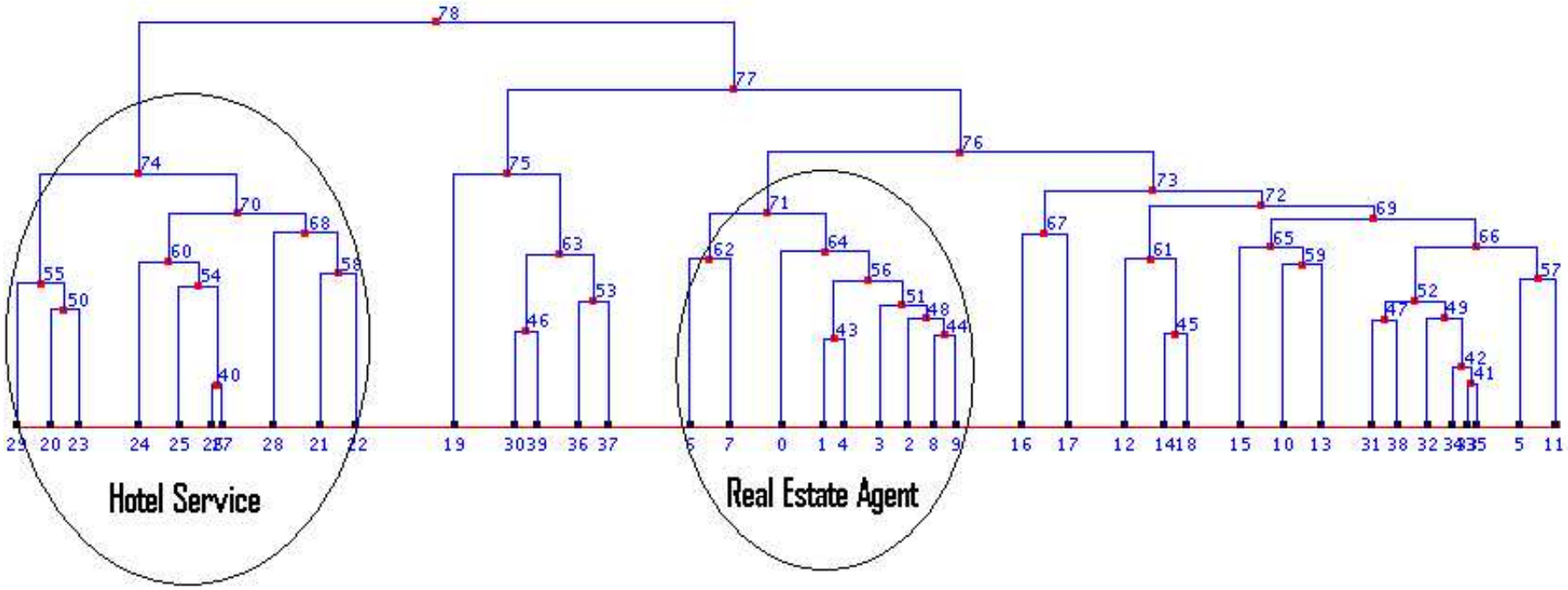


Figure 3.19: Classical AHC on the 40 urban itineraries

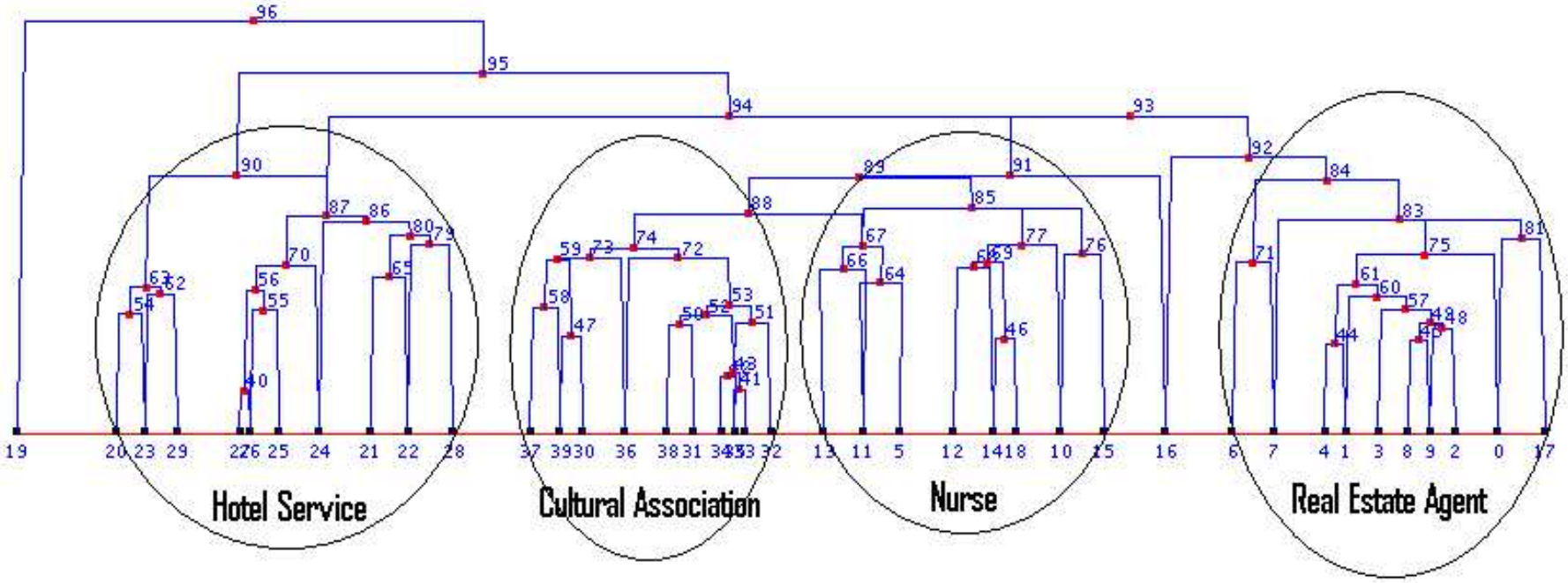


Figure 3.20: 2-3 AHC with intermediate merging (2-3 AHC V2) on the 40 itineraries

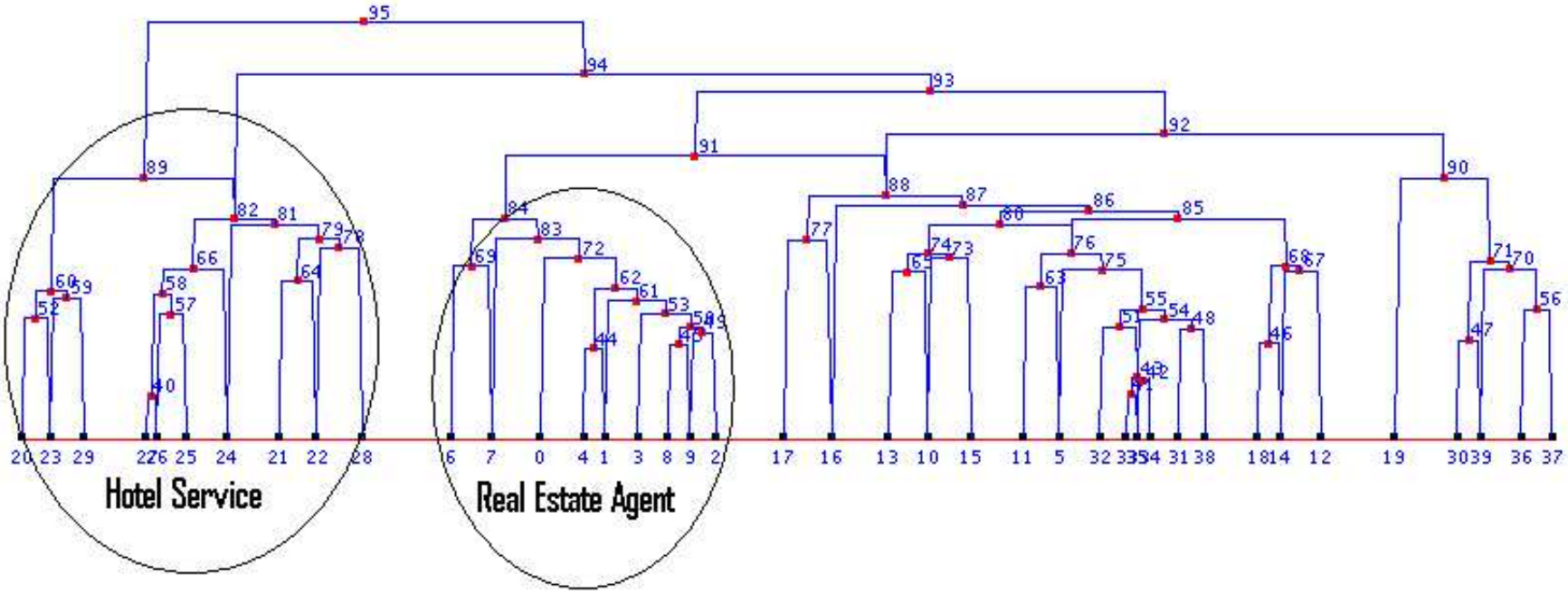


Figure 3.21: 2-3 AHC avoiding blind merging (2-3 AHC V3) on the 40 urban itineraries

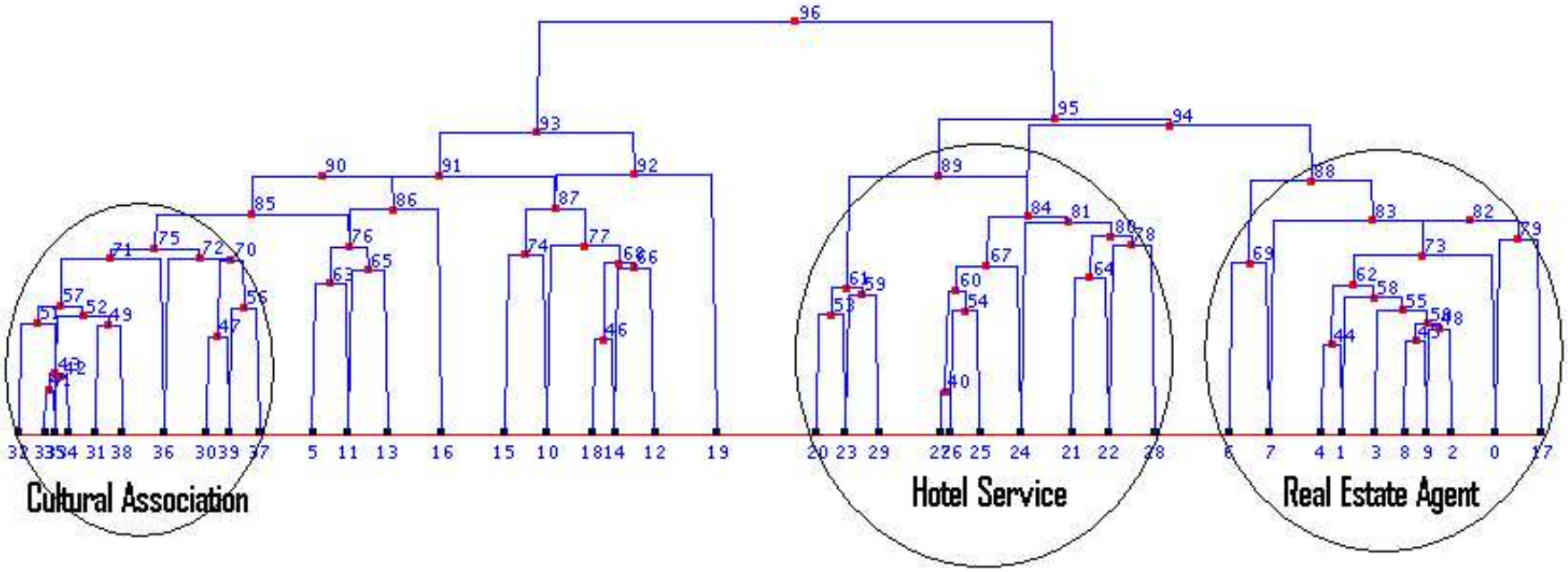


Figure 3.22: Initial 2-3 AHC (2-3 AHC ini) on the 40 urban itineraries

	Nb clusters	Gain nb cl	Stress	Gain Stress
AHC	39		0.241947	
2-3AHC ref V2	55	41.03%	0.112713	*53.41%*
2-3AHC V2	57	46.15%	0.112713	*53.41%*
2-3AHC ref V3	56	43.59%	0.151242	37.49%
2-3AHC V3	56	43.59%	0.151242	37.49%
2-3AHC ini	57	46.15%	0.295758	-22.24%

Table 3.4: Complete link results on the professional itineraries classification

The best classification was obtained using our 2-3 AHC without blind merging avoidance (cf. Figure 3.20) with the biggest Stress gain (53% in Table 3.4) and all four groups identified: real estate in cluster 84, nurse in cluster 85, hotel service in cluster 90 and cultural association in cluster 74. As in the initial 2-3 AHC case, a part of the nurse group was also close to the cultural association group, but this time the nurse group was created. Only four itineraries (10%) were misclassified using the 2-3 AHC, whilst the classification quality is clearly better compared to the classical constructed hierarchy.

To conclude, the 2-3 hierarchies contain more information about the analyzed data than the classical hierarchies (more created clusters, proper intersections, etc.) and can also provide a better data classification for the analyst.

After the analysis performed on these small datasets (Ruspini, itineraries) to reveal the advantages of the 2-3 hierarchies over the classical hierarchies, we analyze next the creation time and the Stress gain of the created structures (hierarchies, 2-3 hierarchies) on larger datasets: on random generated data (cf. Section 3.5.3) and also on a larger real dataset, Abalone [Sam95] (cf. Section 3.5.4).

3.5.3 Artificially Generated Data

After applying the 2-3 AHC algorithms and the classic AHC one on the small Ruspini dataset, we will now perform the same analysis on larger volume artificial generated data to analyze the created structures according to the first two criteria defined in Section 3.5.

For this, we uniformly simulated datasets with the purpose of making sure that a data structure close to the 2-3 hierarchical structure is, if not found, at least better represented better by a 2-3 hierarchical classification than by a classical hierarchical

classification. As we mentioned before (cf. Section 2.2.1), the 2-3 hierarchical structure is a type of parsimonious pyramidal classification, which includes the classical hierarchical classification as a particular case. It is thus probable that a 2-3 hierarchical classification brings an information gain, compared to the classical hierarchical classification, when the dataset has a structure conformed to the general model of pyramidal classification. Such data example, are the data ordered according to a continuum, like the chronological order or the letters order in a genomic sequence.

Therefore, we first generated uniformly n points (25 datasets for each $n \in \{10, 20, 30, 40, 50, 75, 100, 150, ..., 3000)$ according to a sinusoidal curve and using a uniform noise: $y_i = sin(2i\pi/n) + u_i$, with u_i a uniform random variable between 0 and $k \in \{0.02, 0.04, 0.06, 0.08, 0.1\}$.

We also generated points uniformly distributed in a rectangle with $[0.5, 50]$ dimensions. These two types of structure have together the property to distribute the data according to a continuum (sinusoidal curve and line respectively) to which one adds a uniform noise, and consequently are well adapted to a classification whose clusters are total order intervals on E (which is the case of the hierarchies and the 2-3 hierarchies).

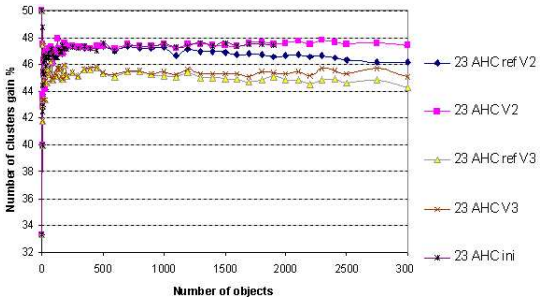


Figure 3.23: CL avg. created clusters number

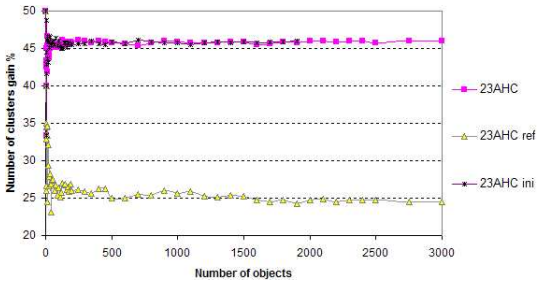


Figure 3.24: SL avg. created clusters number

Firstly, we take a look at the number of cluster created by the 2-3 AHC algorithms compared to the classical AHC one. We present here the results on the rectangle datasets, the results on the sinusoidal datasets were almost identical.

As we can see in Figure 3.23, for the complete link the gain in the number of created clusters of the five 2-3 AHC methods ranged from 32% to 50%. Our 2-3 AHC algorithm with intermediate merging (without integrated refinement and blind merging avoidance) created generally the biggest average number of clusters: 47%. For the single link (see Figure 3.24) the gain ranged from 23% to 50%, while the initial 2-3 AHC algorithm and the 2-3 AHC algorithm without refinement generated the biggest average number of cluster: 46%.

Next, we analyzed the "quality" of each method using the Stress coefficient. For each one of the six executed methods we computed its Stress coefficient, Δ, which

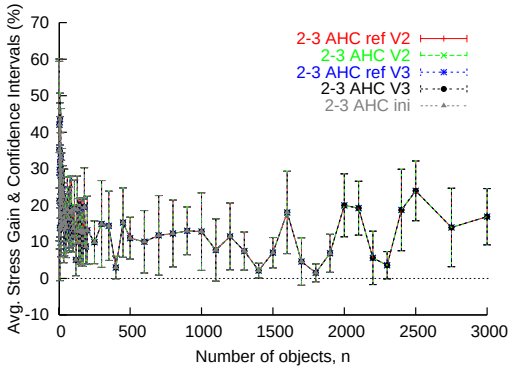


Figure 3.25: SL avg. Stress gain with confidence intervals

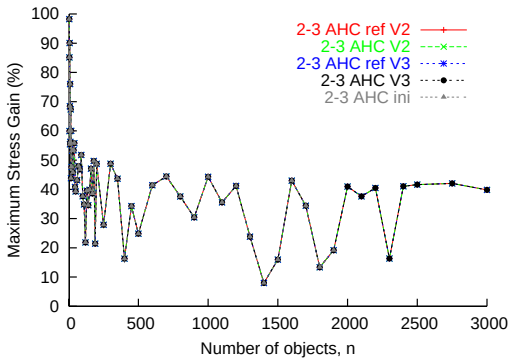


Figure 3.26: SL max. Stress gain

determines how well the initial data are represented by the method. Then, for all 2-3 AHC algorithms, we computed their gain (in %) reported to the classical AHC one: $G = 100(\Delta_{AHC} - \Delta_{23AHC})/\Delta_{AHC}$.

For the single-link, all 2-3 AHC methods had the same Stress gain compared to classical AHC. The average gain varied from 5% to 20% (cf. Figure 3.25) and was very variable with sometimes large confidence intervals (see also Section 2.7.2 for some explications). The maximum gain attained 53% for $n > 50$ (for smaller n, the gain can be bigger: 100% for $n = 3$), while the minimum gain was 0% in this case. All methods had in this case the same gains, which can be explained by the nature of the datasets (continuum) and the particularity of the single-link. More graphics are presented in Appendix C.

Same results variability was obtained for the complete-link: Figures 3.27 and 3.28. One can notice the fact that the gain with our 2-3 AHC algorithm avoiding the blind merging was always positive (5%), while the other 2-3 AHC algorithms had also information loses (negative gain). In this case, the average gain was around 23% but very variable (from 14% to 37%), with an approximate 50% maximum (see Appendix C for more graphics).

3.5.4 Abalone Dataset

After the result analysis on the artificial datasets, we tested our algorithms on a real dataset. A large number of well-known machine learning datasets are available online for machine-learning analysis. The department of Information and Computer Science of the University of California, maintains a repository with such dataset at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/.

For our analysis we chosen from this repository the Abalone dataset [Sam95] which contains 4177 individuals characterized by eight variable (one nominal and seven continuous) such as sex, length, diameter, shell weight, etc. The missing values were removed

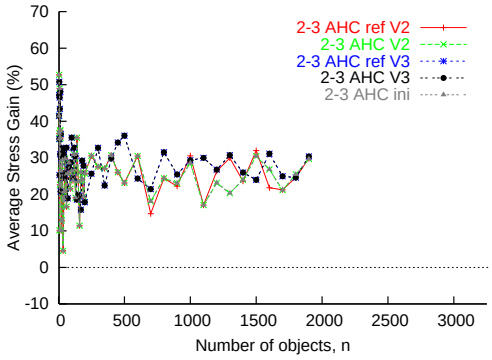


Figure 3.27: CL avg. Stress gain

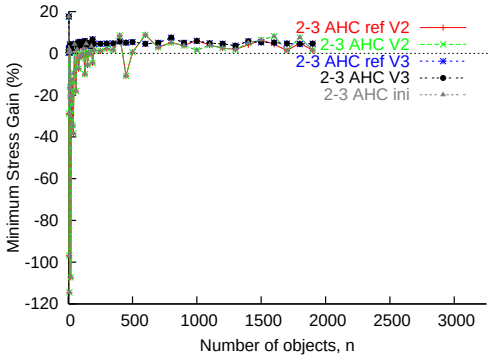


Figure 3.28: CL min. Stress gain

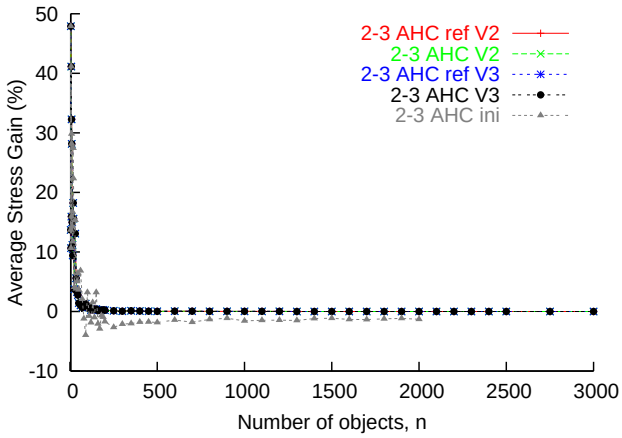


Figure 3.29: Single link's average Stress gain on the Abalone dataset samples

from the original dataset, while the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

To perform our tests, for each $n \in \{10, 20, 30, 40, 50, 75, 100, 150, ..., 3000\}$ we randomly picked n individuals and then computed the dissimilarity matrix (15 iterations for each n) which was used then as the input for the hierarchical algorithms executions.

We present here the Stress analysis results, the gain in the number of created clusters being similar to the one on the artificial datasets.

Very small gains (below 1%) were obtained in the single-link case (see Figure 3.29) with our 2-3 AHC algorithms, while for the initial 2-3 AHC algorithm there were information loses (-3%).

In the complete link case, the 2-3 AHC algorithms avoiding the blind merging were the only one to always provide a positive gain (cf. Figure 3.31), and also the biggest average gain (cf. Figure 3.30. As on the artificial datasets, the gain here was very variable with peaks of 84% for some of the 2-3 AHC algorithms (cf. Figure 3.32).

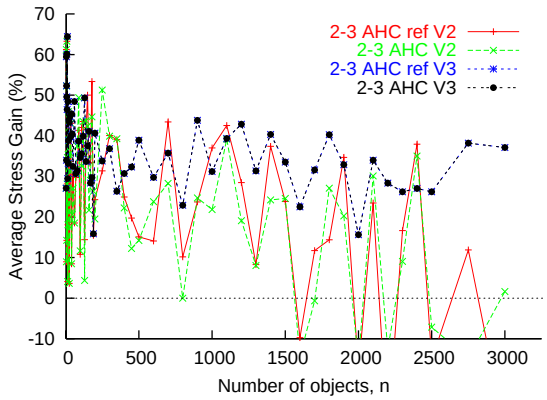


Figure 3.30: CL avg. Stress gain on the Abalone dataset

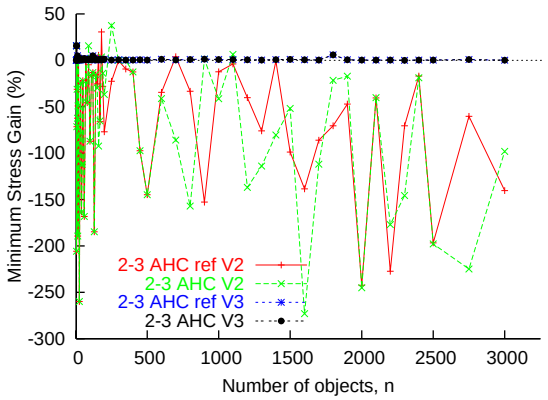


Figure 3.31: CL min. Stress gain on the Abalone dataset

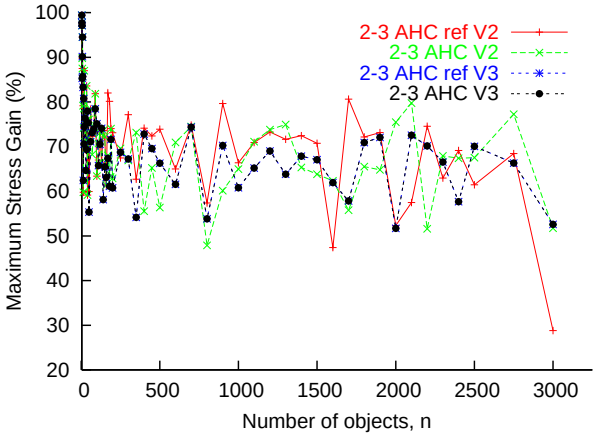


Figure 3.32: CL max. Stress gain on the Abalone dataset

It seems thus that the most suited algorithm is the 2-3 AHC avoiding the blind merging, since it always has an positive gain compared with the other methods. But on new datasets, one can always choose to execute all algorithms if possible, and then based on the Stress coefficient to select the most appropriate one for the results interpretation or the clusters partitioning.

We saw in this section the comparison of our 2-3 AHC algorithms with the classical AHC on different datasets: Ruspini dataset, urban itineraries, generated data, etc. In the next section, a small comparaison between our 2-3 AHC algorithms and an Ascendent Pyramidal Classification (APC) algorithm [EUR] is performed using one of the previously mentioned datasets (urban itineraries).

3.6 Comparison between 2-3 AHC and APC

In this section, we perform a small comparaison between our 2-3 AHC algorithms and an Ascendent Pyramidal Classification (APC) algorithm (see Section 1.2.1 for more details on the APC).

For this, we used the SODAS software [EUR, Did02] in which an APC algorithm was implemented. The last version of the software is called SODAS2, and is a result of the ASSO* Project started in 2001 and aimed at Symbolic Data Analysis. Since the SODAS software is implemented in C++ and our algorithm were implemented in Java, we restrict this comparaison to the resulting stuctures properties. This means that no execution time analyses were performed.

Concerning the algorithms complexities, we note that in the SODAS software a dissimilarity matrix from 800 individuals could not be analyzed using the implemented pyramidal algorithm, whilst our algorithm could analyze matrices of up to 3000 individuals† using the HCT toolbox. However, this is not an reliable complexity comparaison since the implementations are different (different platforms, different codings, etc.). Thus, to compare the two methods we use the created structures: the pyramid and the 2-3 hierarchies on the same dataset.

Since the pyramids can contain a maximal number of non-singleton clusters of $\frac{n(n-1)}{2}$ compared to $(n-1)$ in a classical hierarchy and $\lfloor \frac{3}{2}(n-1) \rfloor$ in an 2-3 hierarchy ($n = |E|$), we chose to analyze a rather small dataset ($n < 50$).

For this analysis we chose the urban itineraries dataset described in Section 3.5.2. We recall that this dataset consist of 40 town-center urban itineraries [Bus05] that where manually generated. They correspond to four types of professional itineraries: *Real Estate Agent* (trips from 0 to 9), *Nurse* (trips from 10 to 19), *Hotel Service* (trips from 20 to 29) and *Cultural Association* (trips from 30 to 39).

The 2-3 hierarchies and the classical hierarchy obtained on this dataset are presented in Section 3.5.2. To compare the 2-3 AHC with the APC, we chose the 2-3 AHC with the best results on this dataset: the 2-3 AHC with intermediate merging (see Figure 3.34).

Using the same dataset as an input (XML file), we then used the SODAS software to construct a pyramid using the APC method. We used the complete-link for both methods executions on this dataset. Figure 3.33 presents the obtained pyramid on the 40 itineraries, whilst Figure 3.34 presents the 2-3 hierarchy.

*Analysis System of Symbolic Official data (`http://www.info.fundp.ac.be/asso/`)

†The tests were done on the same computer: Intel P4 with 512 MB of RAM and Windows XP

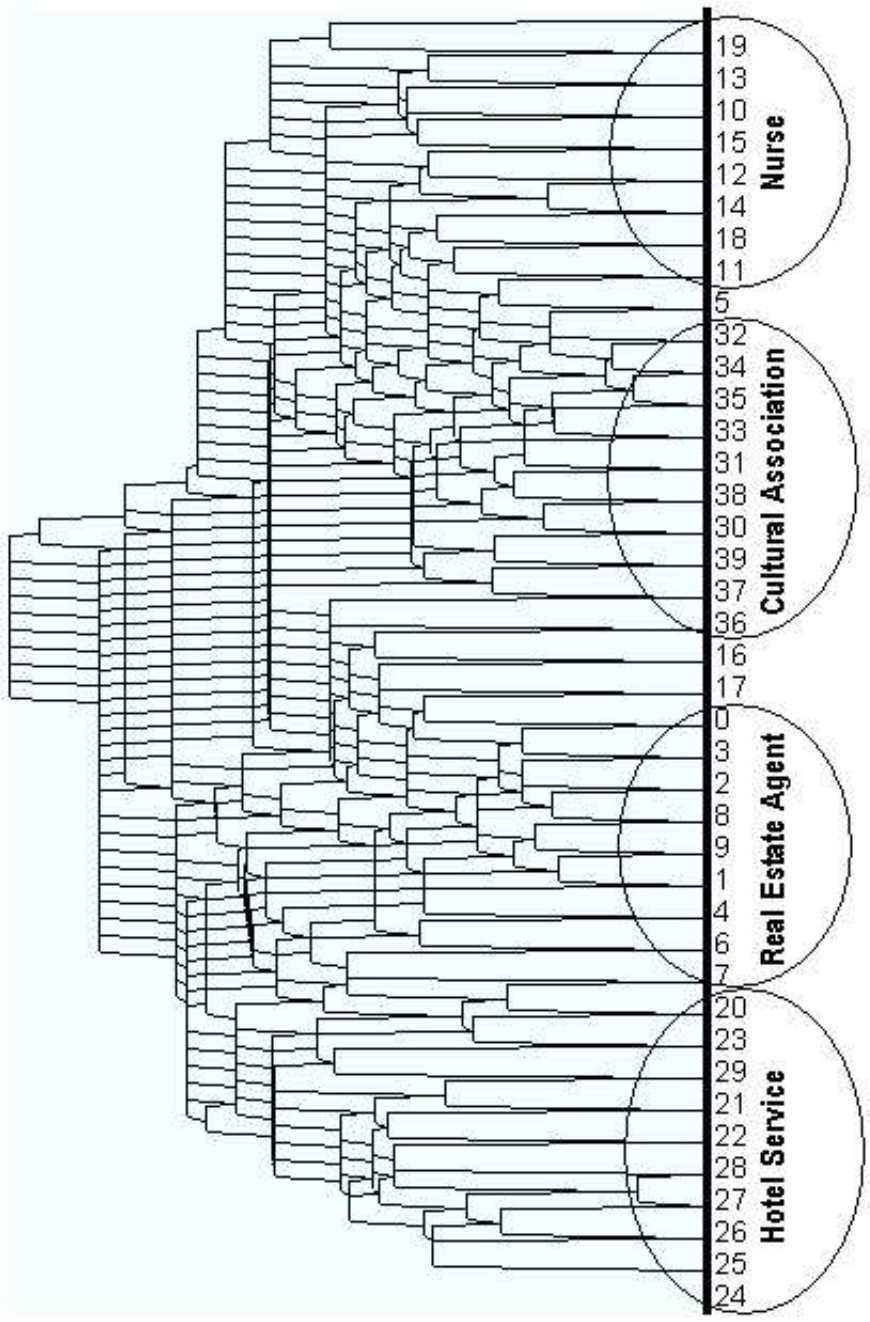


Figure 3.33: Pyramid on 40 urban itineraries

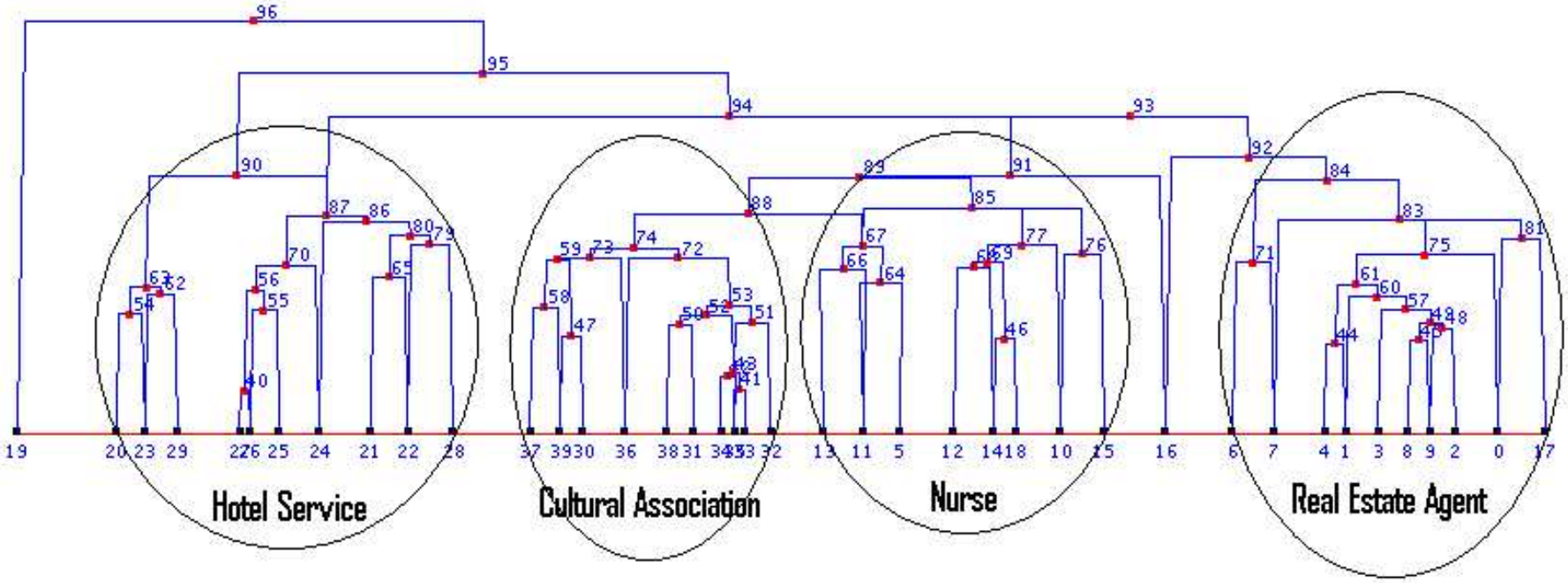


Figure 3.34: 2-3 AHC with intermediate merging (2-3 AHC V2) on 40 urban itineraries

As we can see in the obtained pyramid, it is difficult to identify classes containing different types of itineraries. This is due to the high number of created clusters and their high overlapping degree. In [RD04], authors proposed a method to reduce the number of created clusters in a pyramid to better interpret the obtain results, but this is not yet implemented in the SODAS software. However, the linear order of the trips given by the pyramid separates the analyzed urban itineraries into the four types of itineraries, visible in Figure 3.33.

We also tried to compare the induced dissimilarites matrices of the pyramid and our 2-3 hierarchy, but this was not possible due to wrong computation of the induced matrices in the SODAS software. Using manual generation of induced dissimilarity matrices on small datasets (up to 9 individuals), it appears that the pyramidal classification has better results than our 2-3 AHC: up to 20% Stress gain compared to 2-3 AHC.

To conclude, the 2-3 AHC can be applied on larger datasets and creates less clusters than a pyramid, which can lead to a better visual interpretation of results. But, for a more precise comparison, more tests are necessay between the two methods using similar and updated implemetations: on induced dissimilarities, using same programming language (implementations), etc.

3.7 Discussion and Perspectives

In this Chapter, we used three theoretical properties from the previous Chapter 2 to propose a new general 2-3 AHC algorithm reducing the complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 \log n)$. The main difference between this algorithm and the initial 2-3 AHC algorithm [Ber02d] is the added *intermediate merging* step, which can create a different structure. As a direct consequence of the intermediate merging, the principle of this new 2-3 AHC algorithm is similar to the one of the classical AHC algorithm.

Since our general 2-3 AHC algorithm contains two optional steps, the *integrated refinement* and the *blind merging* avoidance, we practically have four new algorithm variants which can produce different 2-3 hierarchies depending on the currently analyzed dataset.

After presenting a detailed version of our 2-3 AHC algorithm, we proved that it's theoretical complexity is indeed in $\mathcal{O}(n^2 \log n)$. The tests on different datasets (artificial and real) have confirmed the result of our theoretical complexity study and the gain of the 2-3 hierarchies over the classical hierarchies.

Next, we performed a comparative study between the classical AHC, our 2-3 AHC algorithms and the initial 2-3 AHC algorithm:

- First, we analyzed *the number of created clusters* with each 2-3 AHC algorithm compared with the classical AHC one. For any of the 2-3 AHC algorithms, we observed for the complete link an average gain of 47% with an maximum of 50%, whilst for the single link we observed an 23% average gain with a maximum of 45%.

- To *compare the created structures*, we used two types of qualitative analysis: a statistical one, the *Stress* coefficient on the induced dissimilarity matrices and the input matrix, and also a direct result interpretation of the structures on small datasets (Ruspini):

  - For the first analysis, we performed tests on large simulated datasets and also on a real dataset samples. Using the Stress coefficient, the gain was very variable ranging generally from 0% to 43% for the single link. For the complete link, the only algorithm having always a positive Stress gain was our 2-3 AHC algorithm avoiding the blind merging, the rest had sometimes negative gains (loses). The maximum gain reached in this case 84% for the 2-3 AHC algorithms.
  - Using the well-known Ruspini dataset and a small set of urban itineraries, we compared the obtained hierarchies and 2-3 hierarchies and found that extra created clusters of the 2-3 hierarchies provides us with richer informations on the analyzed dataset.

In conclusion, the 2-3 AHC algorithm avoiding the blind merging is the most "stable" one, although its Stress gains were also variable, and should be the one to chose if one wants to assure the construction of a richer 2-3 hierarchy (than the classical hierarchy) on a give dataset.

Future research directions include the definition of other "quality" measures to compare the different hierarchies and 2-3 hierarchies on same datasets. One such measure could be based for example on the clustering accuracy of the 2-3 AHC algorithm compared with the classical AHC or even other clustering methods (neural networks, K-means, etc.).

We also compared our 2-3 AHC algorith with the Ascendent Pyramidal Classification algorithm [EUR]. For this we used the SODAS software [Did02], but due to different implementations a concrete comparison could not be perform. This is subject of other future work.

Chapter 4

Toolbox for Hierarchical Clustering Methods and CBR*Tools Integration

In this Chapter we present the object-oriented model that we proposed for the classical AHC, the initial 2-3 AHC [Ber02d] and our 2-3 AHC algorithm. Its integration in two different software developed in our team is described.

This object-oriented model was used:

- first for the design and implementation of the `Hierarchical Clustering Toolbox` that we developed. This graphical toolbox is designed to ease the results visualization and interpretation for the previously mentioned algorithms. The toolbox is described in Section 4.2.

- to integrate our 2-3 AHC algorithm in the existing *CBR*Tools* framework for Case Base Reasoning (see Section 4.3.1). The main motivation here was the use of these algorithms in the Be-TRIP mobility recomender system (based on CBR*Tools) that we specified (cf. Appendix F).

After the description of our object-oriented model in Section 4.1, we detail in Section 4.2 the graphical toolbox that we developed. Section 4.3 presents the integration of the model into the existing CBR*Tools framework.

The applications and future uses of our 2-3 AHC algorithm using the two aformentioned software are then presented in Section 4.4 before concluding in Section 4.5.

4.1 Design of an Object-Oriented Model

We based our implementation of the algorithms on a reusable model which can be easily integrated into different applications and libraries, like the Case-Based Reasoning

framework CBR*Tools [Jac98] developed in our team.

CBR*Tools is an object-oriented framework for Case-Based Reasoning (CBR). It provides a basic *reusable* CBR framework that supports the development of CBR applications [Jac98]. It can be especially used for problems addressing behavioral situation retrieval and indexation.

Different applications using the CBR*Tools framework have been developed in our team:

- Plant nutrition control application (in collaboration with INRA Sophia-Antipolis);
- *Broadway-Web* [TJK99]: recommender systems supporting internet browsing;
- *HERMES* [KTP97]: for argumentation in collective decision making;
- *Be-TRIP* [CGT04, TCG04]: mobility recommender system that we recently designed. This recommender system is not implemented yet, Appendix F contains detailed description of the proposed architecture (in french).

To integrate our algorithm in CBR*Tools, we adopted the same design approach, based on the concepts found in a framework (e.g. *hot-spots* [Sch97]). Therefore, we first make a brief introduction of the frameworks and their concepts.

4.1.1 Notion of an Object-Oriented Framework

The concept of object-oriented frameworks has been introduced in the late 80's and has been defined as "a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes" [JF88]. Thus a framework is much more than a software library. A library defines a set of classes that may be reused independently or in very small groups. On the other hand, the goal of a framework is to capture a set of concepts related to a domain and the way they interact. In addition a framework is in control of a part of the program activity and calls specific application code by dynamic method binding. A framework can be viewed as an incomplete application where the user only has to specialize some classes to build the complete application.

The design process is centered on the identification of *hot-spots* [Sch97]. Hot-spots are well defined features of the framework that can be customized for a specific application by specialization (white-box hot-spot) or composition (black-box hot-spot). At the beginning a framework only defines white-box components and a more mature framework will also provide black-box components. The hot-spots are usually created by using *design patterns* which provide typical design solutions and improve the framework documentation.

Therefore designing a framework is a complex task with several issues that have to be overcome and the framework requires a good documentation to be actually reused. However, the framework approach is very appealing. Frameworks allow the reuse of both code and design for a class of problems, giving the ability to non-expert to write complex applications quickly. Frameworks also allow the development of prototypes which could be extended further on by specialization or composition. A framework is more difficult to understand than an application or a class library, but once this step is done, the framework can be applied in a wide spectrum of context, and can be enhanced by the integration of new components.

4.1.2 Object-Oriented Model

The current design of the 2-3 AHC algorithm implies the two following hot-spots:

- **DistanceMeasure**: offers support for implementing the distance metric used during the initialization phase (distance matrix creation). Default implementation for this hot-spot is provided by the PondEuclidDistance specialization, but others like the *Minkowski distance* that generalize the first one, can be used.
- **ClusterDisssimilarity**: offers support for implementing the dissimilarity (linkage) between clusters, used during the merging phase. Implementation for this hot-spot include: SingleLinkDissimilarity and CompleteLinkDissimilarity specializations (others like the *average linkage* or the *Ward criterion* can be used).

The object-oriented model of our index is presented in Figure 4.1. Its design takes into account the future integration into CBR*Tools: the HAC23Index, HAC23Index_ini and HACIndex classes specialize the SimpleIndex class (see Section 4.3.2).

We implemented three hierarchical agglomerative methods: the classical AHC algorithm **HACIndex**, the initial 2-3 AHC algorithm **HAC23Index_ini** and our 2-3 AHC algorithms **HAC23Index**. To execute different variants of our 2-3 AHC algortihm, one must simply define the three parameters in the HAC23Index class such as _integratedRefinement or _avoidBlindMerging.

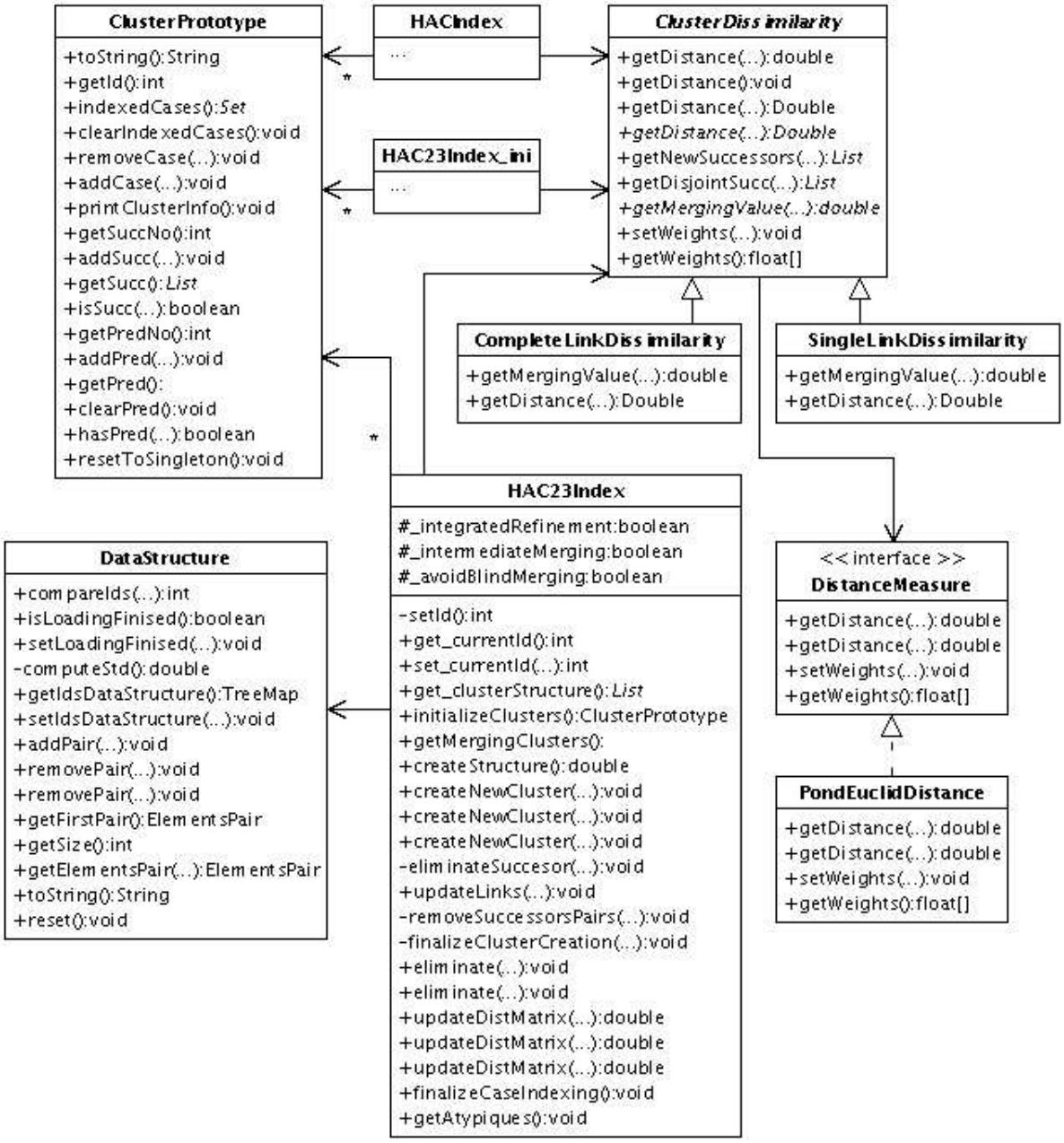


Figure 4.1: Object oriented model of the agglomerative hierarchical algorithms

4.2 Toolbox for Hierarchical Clustering Methods

To better visualize, compare, interpret and extract information from the created hierarchies and 2-3 hierarchies, we designed and implemented the `Hierarchical Clustering Toolbox` (HCT). It uses the object-oriented model of the algorithms from the previous Section 4.1.

The toolbox was developed in Java and can be run as a stand-alone application or as a applet in a Web browser via the internet*. It contains:

- modules for:
  - object-oriented models of algorithms;
  - dissimilarity matrix generation from different data inputs;
  - results analysis and comparaison.
- a Graphical User Interface (GUI) for:
  - input selection and representation;
  - method selection and execution;
  - results representation.

To use the toolbox, there are three (successive) main tasks that can be performed: data selection and representation (see Section 4.2.1), method selection and execution (see Section 4.2.2) and results visualization and analysis (see Section 4.2.3).

These tasks can be performed using the three application's menus or the toolbars present in the different tabs. The tabs of the toolbox, *Input Data Graphical Representation* (see Figure 4.4), *Output Structure Graphical Representation* (see Figure 4.6), and *Analysis* (see Figure 4.8), correspond to the general main steps of a data analysis for the hierarchical methods and are presented in details in the next three Sections.

4.2.1 Data Selection and Representation

When selecting the analyzed data, various dialogs of the toolbox can be used to load/generate the input data. Thus, the analyzed data can be:

- artificially generated in a 2D rectangle (used mostly for tests);
- loaded from a file. XML, SDS [EUR] and plain text files are supported;
- extracted via SQL queries from a database (see Figures 4.2 and 4.3).

*Running the application as an applet is not recommended on large datasets due to the JVM memory limitations and the large size of the dissimilarity matrix, $\mathcal{O}(n^2)$.

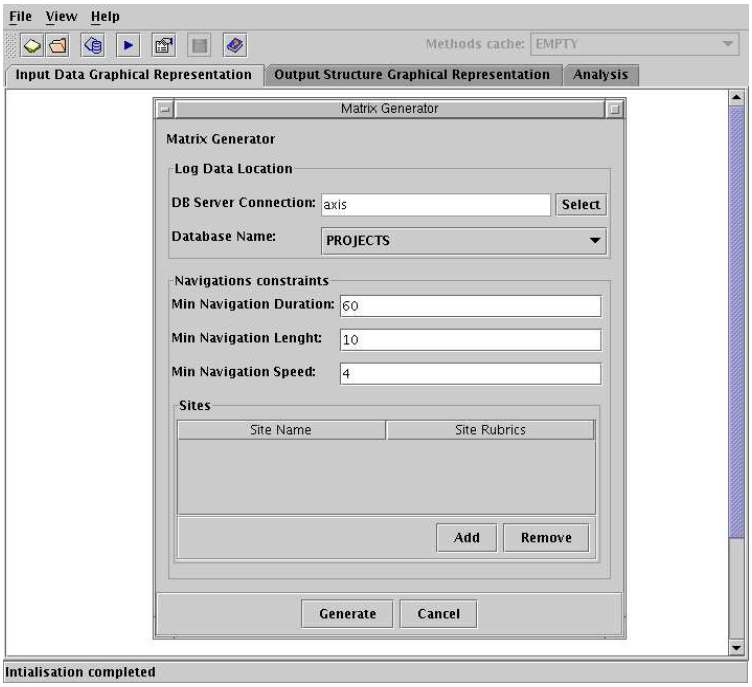


Figure 4.2: HCT: Data matrix generation from DB

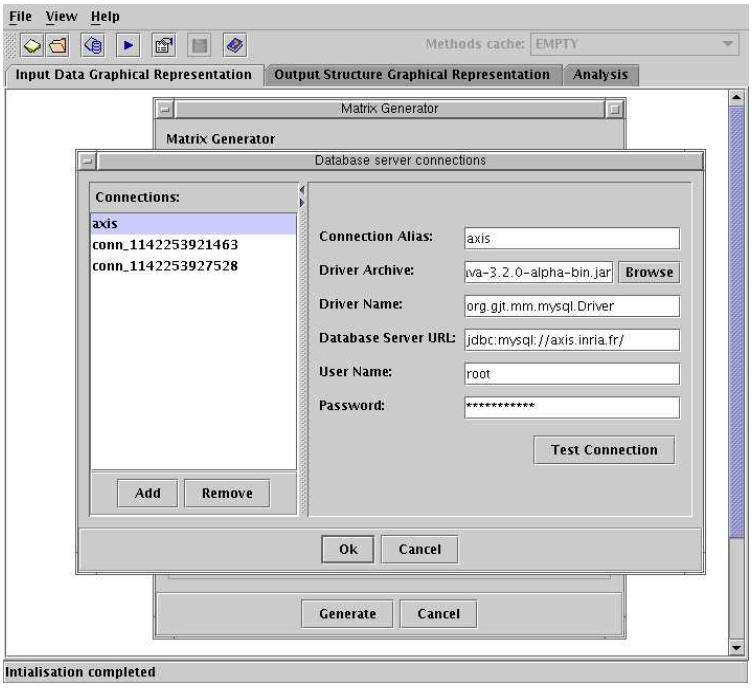


Figure 4.3: HCT: DB server selection

The later functionality was created having a specific purpose in mind: to generate and extract dissimilarity matrices on Web user sessions (see Chapter 5), but it can be easily extended to general data extraction queries from databases. This can be done using a more general DB Explorer feature (as in [Tan05a]), in which custom queries for selecting data can be made after exploring the available data in the selected DB.

After this, a two dimensional representation of the input data is displayed (when possible). This is done on the first tab of the toolbox, Input Data Graphical Representation, as in Figure 4.4. In the case from Figure 4.4, the Ruspini dataset [Rus69] is used as input and represented graphically.

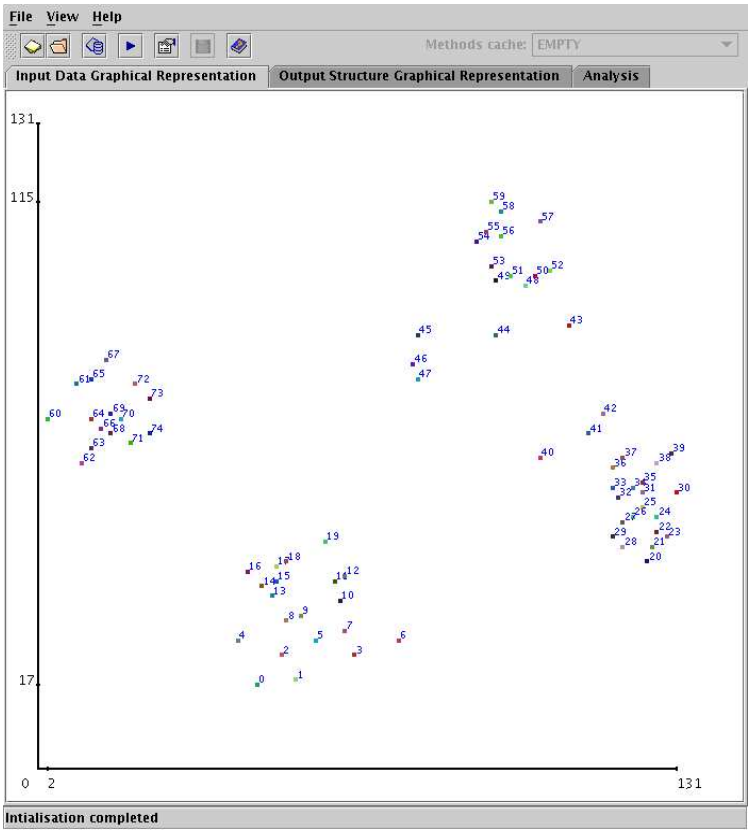


Figure 4.4: Hierarchical Clustering Toolbox: Input Data Representation

4.2.2 Methods Selection and Execution

After selecting the input data, the method selection dialog is displayed, and we have a choice between various algorithms (cf. Figure 4.5):

- the classical AHC algorithm;
- the initial 2-3 AHC algorithm;
- the four variants of our 2-3 AHC algorithm:
  - with intermediate merging and integrated refinement (denoted *HAC23 with refinement V2*),
  - with intermediate merging and no refinement (denoted *HAC23 without refinement V2*),
  - with intermediate merging, integrated refinement and avoiding the blind merging (denoted *HAC23 with refinement V3*),
  - with intermediate merging, no refinement and avoiding the blind merging (denoted *HAC23 without refinement V3*).

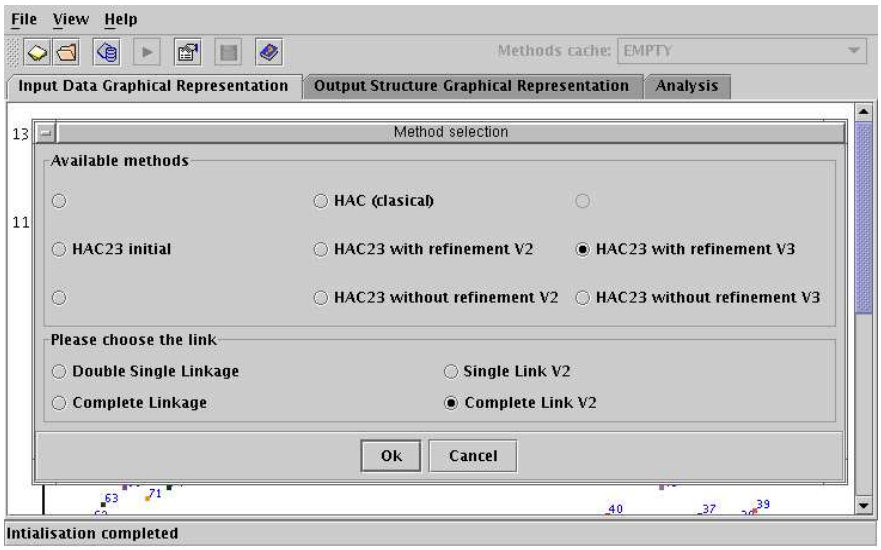


Figure 4.5: HCT: Choosing the linkage and the algorithm to execute

All algorithms can be executed successively on the same dataset in order to compare the structures later.

4.2.3 Results Visualization and Analysis

After the execution of an algorithm, the created structure (hierarchy or 2-3 hierarchy) is displayed in the second tab (Output Structure Graphical Representation) of the toolbox (see Figure 4.6). This tab is used for visual result interpretation.

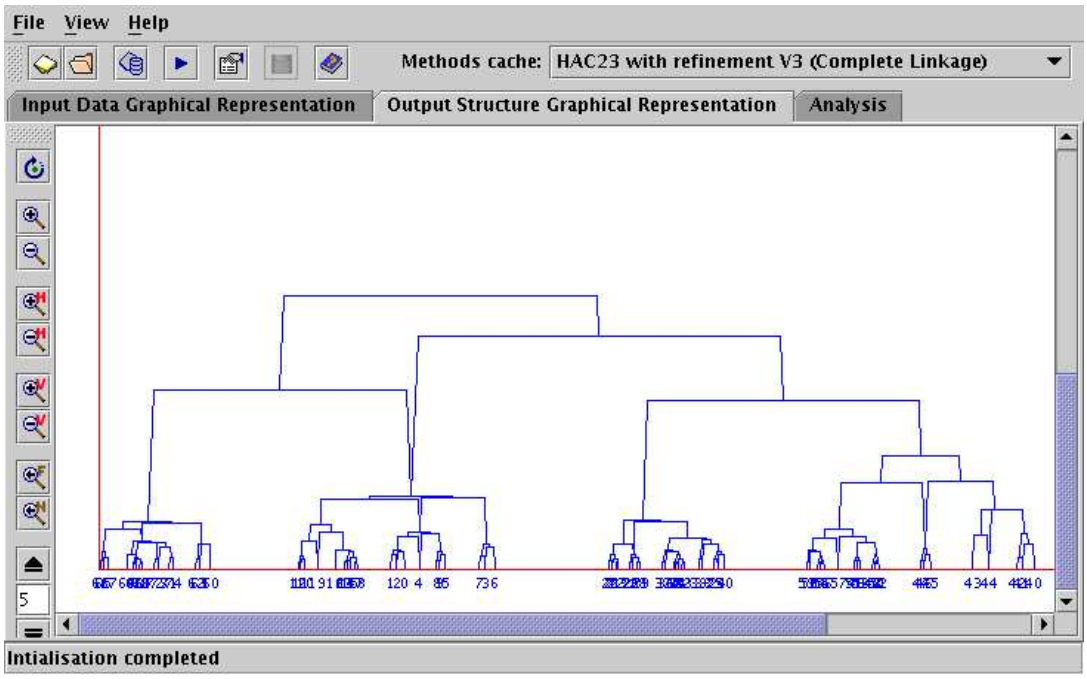


Figure 4.6: HCT: Output Structure Graphical Representation

Here one can analyze the different structures created on the same input dataset using the *Method Cache* dropbox in the upper right corner of the toolbox. Thus differences between the classical hierarchies and the 2-3 hierarchies can be revealed and interpreted.

The different implemented features ease the results visualization and interpretation: cluster rotation (for the non-properly intersecting ones), cluster information, different types of zoom, rotation of the structure (horizontal and vertical display), etc.

The overlapping obtained by selecting an indexing level can be done by direct selection on the structure or by manually entering the desired level (cf. Figure 4.7). The informations of the resulting overlapping are displayed internally in the toolbol and/or externally in a DVI viewer such as *xdvi* or a Web browser. This information can then be saved as HTML, LaTeX (tables) or plain text.

Navigation between the clusters successors and predecessors can be easily made after partitioning or directly in the represented structure using the IDs links.

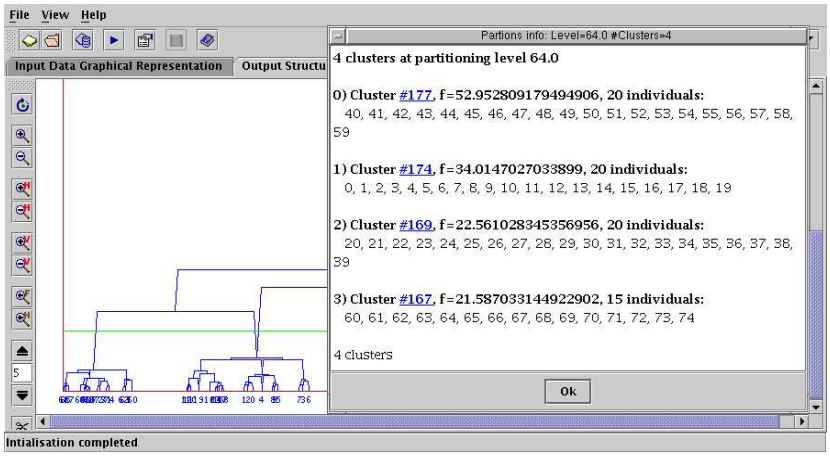


Figure 4.7: HCT: Selection of a clustering partition

To compare the input dissimilarity matrix and the induced dissimilarity matrices, the third information tab of our toolbox, Analysis, can be used (cf. Figure 4.8).

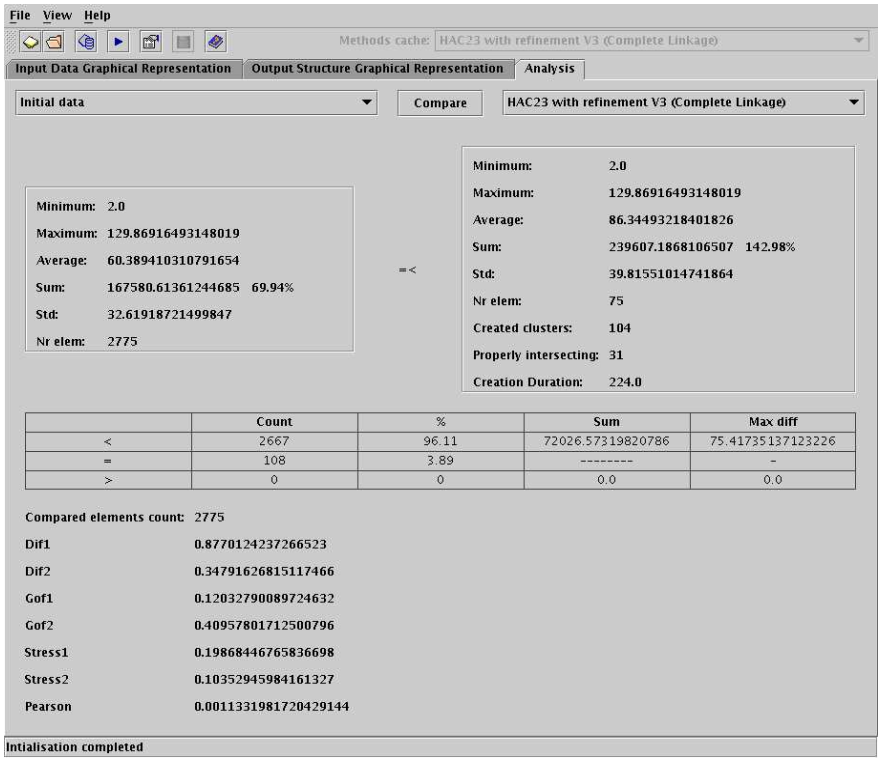


Figure 4.8: HCT: Induced dissimilarites and initial data matrices analysis

Some statistical information on the compared dissimilarity matrices are presented,

such as the minimum, maximum, standard deviation, etc. along with additional informations on the structure: the creation time (if is an induced matrix), the number of created clusters and properly intersecting clusters, etc.

Also some matrix correlation coeficients (see Section 3.5) are computed and displayed: *Stress*, *Pearson*, etc. These can be used as a "quality" indicator by comparing the induced matrices to the initial dissimilarity.

Using our toolbox, the maximum attained number of analyzed elements was 3500, but during the repetitive tests from Section 3.4.3 the maximum number was 3000.

4.3 Integration of the 2-3 AHC in the CBR*Tools Framework

Here we will discuss the different aspects related to the conception and implementation of our algorithm as an indexing method and his integration in the object-oriented framework CBR*Tools* developed in our team.

The purpose of this integration is to use our method as an off-line clustering technique for prototypes (profiles) creation and as an on-line indexing method for case retrieval and case retaining (see Figure 4.9). This can be then used in the Be-TRIP recommender system (cf. Appendix F) to construct for example itineraries prototypes (see Section 3.5.2) which helps the recommendation process.

4.3.1 CBR*Tools: a Framework for Case Based Reasoning

Generally speaking, Case Based Reasoning (CBR) [JF88] is a problem-solving method based on the reuse of cases. A case basically represents a problem situation, the solution that has been applied (or a way to compute it), and sometimes its evaluation. Cases must be structured and indexed into a memory in order to be reused when similar problems are encountered.

The first step of the reasoning (cf. Figure 4.9) is the retrieval through indexes of relevant cases which are somehow similar, or match partially the current problem situation. The goals of others steps are the reuse of the past solution by adaptation, the evaluation of the proposed solutions and finally the learning of this new experience in the memory for future reuse.

*http://www-sop.inria.fr/axis/cbrtools/

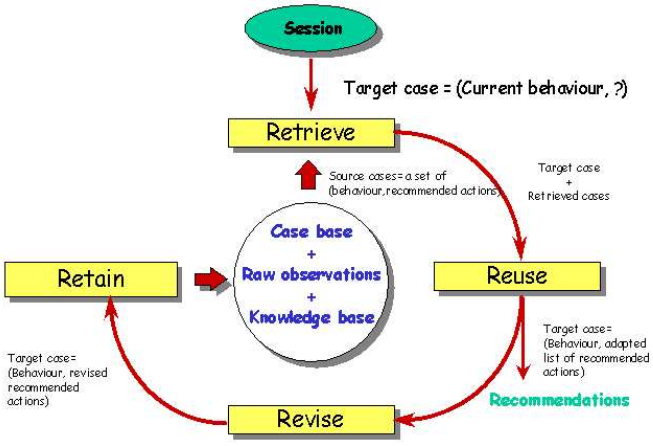


Figure 4.9: Case Based Reasoning

The CBR*Tools framework developed in our team is implemented in Java and contains five basis packages with a total of 246 classes. As we said before, the CBR*Tools framework is based on the use of *hot-spots*, which are mainly structured according to two criteria:

- **Specificity**: Two levels of specificity are identified: the first one called "core" gathers the general hot-spots related to the CBR and the second one called "time" gathers the additional or specialized hot-spots, necessary for the indexing by behavioral situations.

- **Axes of variability**: An axis of variability defines a dimension in which the various applications require flexibility and are likely to vary. Three axes of variability were identified each one gathering a set of hot-spots: *reasoning management*, *cases representation* and *memory organization*.

4.3.2 HAC23Index Integration for Memory Organization

The design of our HAC23Index concerns the third axis of variability represented by the *memory organization* which contains the source cases and organizes them according to the data structures used (see Figure 4.10). Thus we have here two principal sub-tasks: the organization of the cases (physical storage of the cases) and the organization of the indexing (defining indexes which will make it possible to find the cases at the time of the research phase).

Various indexes are usually used (hot-spot Index): linear organization, discriminating tree built a priori or by induction, neural networks. Moreover, it is necessary to be able to make evolve these indexes during the cycle of life of the system in order to include new knowledge or to deal with the number growing of the cases added

in memory (incremental aspect). It is thus necessary simultaneously to define several strategies of indexing to evaluate them and make them evolve (hot-spot IndexBase). Lastly, the indexes can take into account only part of the indices of the cases (hot-spot IndiceFilter) and turn over various types of information (hot-spot IndexResult): set of cases, set of prototypes, detailed analysis of indices.

In this context the HAC23Index index* is integrated into the CBR*Tools framework in order to create prototypes (clusters) of cases with a strong similarity. Next, these clusters can be used in the retrieve (search) phase in order to return the similar cases to the target one. The main drawback of this approach is the time required to construct the dissimilarity matrix and the hierarchy if a new case is to be added to the case base. In order to avoid this, the clusters are obtained in an off-line analysis and then used to classify a new case, which can be added to the closest cluster found if it is "close"† enough or a new cluster containing this case can be created if it is an "atypical" case. This is known as *incremental clustering* [SBK01] which makes the new case classification and the cluster updating real-time possible. Thus we can make the index an incremental one without having to rebuild the distance matrix and the whole 2-3 hierarchy. Updating the structure when a new case arrives [ZRL96, BHT05] could also be used to make our case indexing incremental.

The index could be used in two different ways: the whole hierarchy created can be used to "search down" similar cases starting from the root or a partitioning of the initial dataset can be selected by "cutting" the hierarchy according to a given condition and the clusters obtained to be used to classify the new case. In our implementation we have use the later and as a "cutting" condition a given threshold. The cutting is performed when there is a variation in the clusters dissimilarity superior to the given threshold, thus it is not necessary to have a strict indices. The problem in this case is the specification of this threshold, which implies apriori knowledge on the analyzed data. This could be avoided by an initial specification of the number of desired clusters, but results can be unexpected.

These various indexes from CBR*Tools make use in particular of measurements of similarity (hot-spot Similarity) making it possible to compare the cases sources with the target case. These measurements turn over evaluations (hot-spot CmpValue) which can be of various types: generally a factor between 0 and 1, or 0 and 100 or a couple possibility/necessity. These results are then ordered according to an associated relation of order (hot-spot CmpValueOrder). See also Appendix D.

Our HAC23Index index is a part of the memory organization axe of the CBR

*Using same reasoning the classical AHC and the initial 2-3 AHC implementations can be integrated into CBR*Tools

†Generally if a distance threshold condition is satisfied

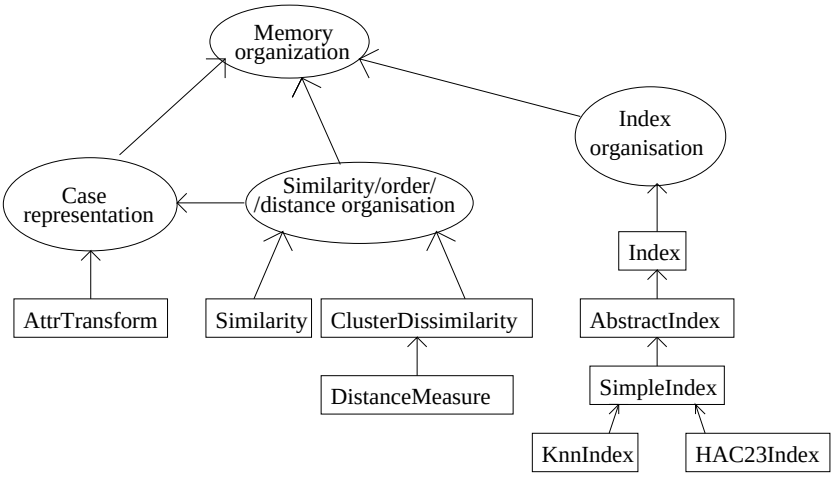


Figure 4.10: HAC23Index hot-spots and their integration in CBR*Tools

framework, and extends and partially implements the specifications offered by the SimpleIndex hot-spot. The major part of its functionality relies on the createHierarchy and an added search method. The former constructs the 2-3 hierarchy after the batch indexing process of the case base was done. That means that there are previously stored patterns in the case base which have to be indexed during the initialization of the application. The search method has a role, as the name states, in the pattern retrieving step of the CBR process.

The HAC23Index is used as a kind of pre-indexing or filtering mechanism, its output being furthermore adapted by an attached linear index like the KnnIndex (see Figure 4.10). So in case of an actual new index implementation, we can use as a start point the LinearKnnIndex or the HAC23Index, depending on the new index's functionality one might want to offer.

In order to point out the basic steps for adding and implementing new indexes to the CBR tools we remark the different knowledge levels of a certain hot-spot, required to use, extend or implement it. In this direction, we can distinguish three levels :

- a specialization and extending of the hot-spot's specification, which needs a complete understanding of it.

- the hot-spot instantiation requires only the knowledge of its parameters and their meaning.

- in case of automatic use of the hot-spot, it could remain transparent to the programmer.

With respect to the hot-spots in Figure 4.10, depending on the application and the chosen case structure, we can provide further specialization for the AttrTransform hot-spot, beside the default implementation in BasicAttrTransform.

An example of the use of HAC23Index as a part of the CBR*Tools is given in Appendix D. The presented CBR application is used to determine the risk factor for car insurances.

4.4 Applications

Based on the two softwares previously presented (HCT and CBR*Tools), our 2-3 AHC algorithms were used in different applications from three fields: Web Usage Mining, Document Clustering and Transport. Three out of the seven applications presented below were made by others members of the AxIS team.

With the first applications from the WUM and Document Clustering fields detailed in the next two Chapters (Chapter 5 and Chapter 6), we will summarize only the tourism applications and the work performed in this context.

- ***Web Usage Mining*** (using HCT):
  - **clustering INRIA's research teams using groups of Web pages and users navigational behaviours [CT04, CT05], presented in Chapter 5**;
  - clustering of search engines keywords extracted from accessed URL referrers [TMT06];
- ***Document Clustering*** (using HCT):
  - **XML documents classification using structure and content mining, presented in Chapter 6**;
  - XML documents classification using structure mining via sequential patterns extraction [GMT05];
  - Sanskrit documents classification [Tan05b] in order to create a critical edition*;
- ***Transport***:
  - **validation of urban itineraries classification (cf. Section 3.5.2) using our 2-3 AHC**†.

    For this, our toolbox was interfaced with a C++ library developed by Benomad‡. This library provides geographical information from a GIS§ map,

*Co-supervision with B. Trousse of S. Tandabany's internship
†Via the co-supervision with B. Trousse of R. Busseuil during his internship [Bus05]
‡`http://www.benomad.com/`
§Geographical Information System

which can be used to create itineraries or to extract an itinerary's characteristics. A Visual .NET application using the Benomad library and our visualization toolbox was developed by R. Busseuil [Bus05] (cf. Figure 4.11). Using this application, the manually generated itineraries could be then easily clustered using our Hierarchical Clustering Toolbox (see also Section 3.5.2).

– **the mobility recommender system, Be-TRIP** [CGT04, TCG04].

As mentioned in Section 3.5.2, the purpose of the urban itineraries clustering was to validate the use of our 2-3 AHC algorithm as a case indexing method for our Be-TRIP mobility recommender system that we specified (see Appendix F). To use our 2-3 AHC algorithms in Be-TRIP, we first integrated our object-oriented model in the Case-Based Reasoning framework CBR*Tools [Jac98] (see Section 4.3), on which the Be-TRIP system is based.

The implementation of Be-TRIP and the validation of 2-3 AHC as an indexing method, are to be done in the future by the AxIS team.

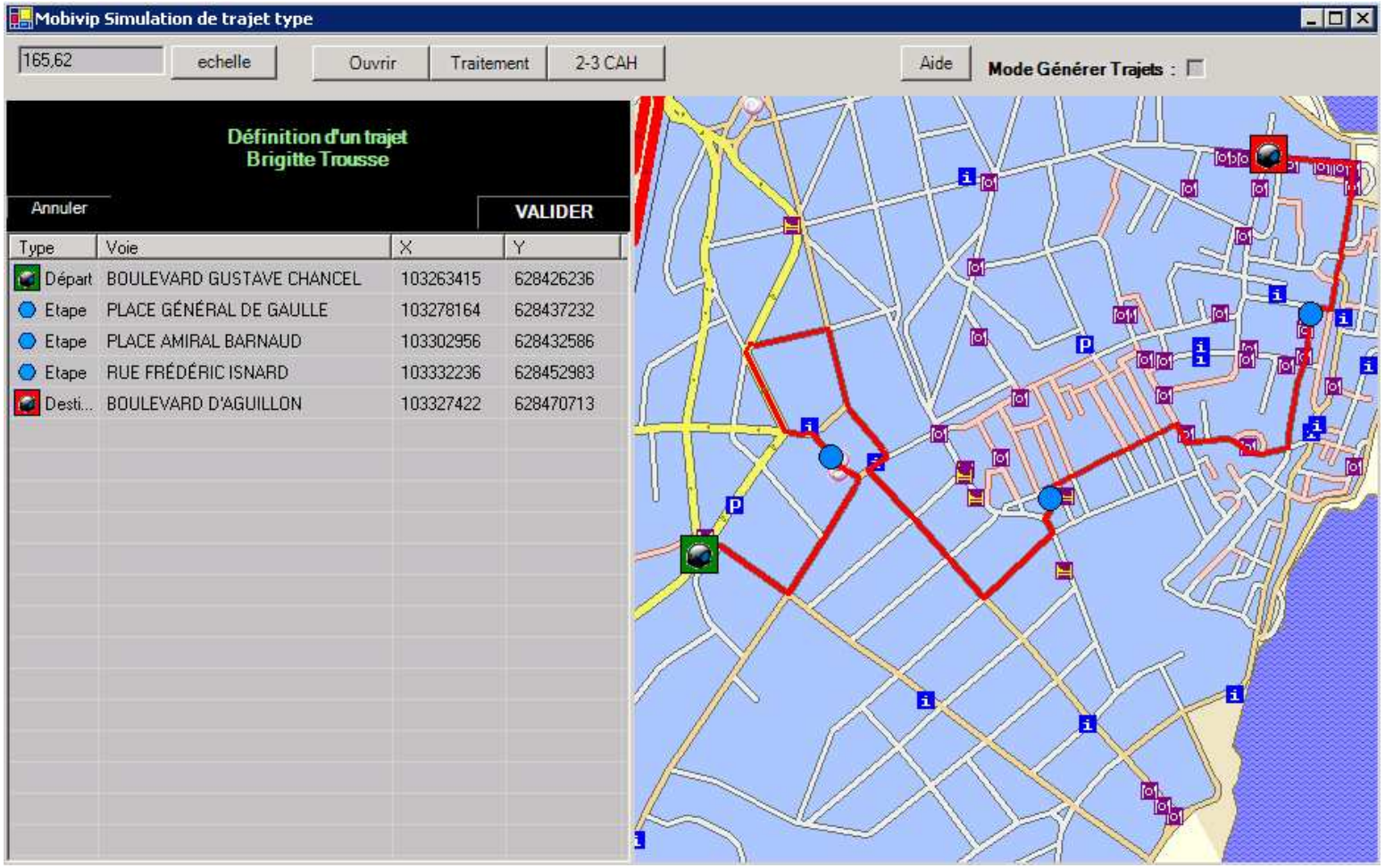


Figure 4.11: Urban Itinerary Creation and Visualization

4.5 Discussion and Perspectives

In this Chapter we presented the object-oriented model of our 2-3 AHC algorithm, and its implementation, which is the first implementation of an 2-3 AHC algorithm. The implementation was performed in Java. Almost identical models were used for the classical AHC and the initial 2-3 AHC algorithms implementations.

To better visualize and interpret the results, i.e. the classical hierarchies and the 2-3 hierarchies, we created the `Hierarchical Clustering Toolbox`, also implemented in Java.

The toolbox can be used to generate the input dissimilarity matrix or to load it from different sources. The structures can be then represented graphically for result interpretation and different indices (Stress, Pearson, etc.) can be used to analyze the "quality" of the created structure.

Currently, we are integrating the data selection from DB in a more general DB Explorer feature in which custom SQL queries for selecting data can be executed after exploring the available data in the selected DB. Also different comparison indices on the created structures and the initial data can be implemented and integrated in the Analysis tab.

We also integrated our object-oriented model of the 2-3 AHC algorithm into the Case-Based Reasoning framework, CBR*Tools [Jac98]. Our algorithm was integrated as an indexing method for the case base organization an used in the retrieving phase of the CBR cycle. Unfortunately, due to its high complexity, only off-line analysis can be performed so far.

Future work regarding the CBR*Tools integration includes the study of an automatic partitioning criteria and also an incremental feature for the hierarchical algorithms to allow on-line analysis.

Part II

Study of applying the 2-3 AHC in the Web Mining and Document Clustering Fields

Introduction

This second part of the thesis concerns two studies of applying the 2-3 AHC in the fields of Web Mining and Document Clustering. These two fields are part of the main research areas of our AxIS* team, and also correspond to very active research fields. We also saw in Chapter 1 that the hierarchical ascending classification methods (AHC, 2-3 AHC, APC, etc.) are rarely or never applied in these research fields.

Therefore, we analyzed the applicability and potential of our 2-3 AHC method in these two fields and also compared it with the classical AHC or other methods ([GCGR+04]) when applied on same data.

The Web Mining can be broken in three categories:

- *Web Content Mining*: discovering and organizing Web based information;
- *Web Structure Mining*: used to examine data related to the structure of a particular Web site;
- *Web Usage Mining* [CMS97] (WUM): Web resources clustering, discovering sequential patterns [TT01], associations rules [AS95], classification, etc..

The most used techniques applied to Web usage data, obtained from Web servers access logs, are: *statistical analysis*, discovering *association rules* [Bar01, AS95], mining *sequential patterns* [SA96, AS95, TT01, MJHS96, ZXH98, MPC99, PHMAZ00, Tan05a] and *clustering* of Web usage data.

As concerning the clustering here there is a big interest in document clustering (organizing and managing the groups of related URLs based on their page content) [Wil88, KR90]. The formed clusters of documents can be used for document management in electronic commerce (customer targeting) [SCH+01]. Also document clustering is explored as an alternative method of organizing retrieval results in search engines [ZE98].

Clustering Web users, i.e., grouping the users into clusters based on their common properties is done on users sessions which usually are modeled as vectors. Each element

*http://www-sop.inria.fr/axis

of the vector corresponds to a value of a feature such as the hit-count for a Web page. Then different clustering algorithms can be applied to discover the user profiles [SBK01, NK02]. In [YKM99] the clustering of the Web users is based on their access pattern (browsing activities) which are organized into sessions, then generalized according to the page hierarchy and afterwards clustered using an incremental hierarchical clustering method, BIRCH [ZRL96]. The same method is used in [FSS00] to cluster generalized Web user sessions obtained by means of induction based on the attributes.

The user clusters obtained by applying different clustering techniques to user sessions [Ben00] are used for classification of new sessions (users) and possibly in an update phase of the clusters (incremental learning) by adding these new sessions to the corresponding cluster (*incremental clustering* [SBK01]).

The Web usage data used during the Web Usage Mining (WUM) process are generally the users' navigational paths gathered in Web server logs, sometimes correlated with informations from the other Web Mining processes, e.g. the site structure.

Analyzing the Web users behaviour can be used by webmasters in the Web site(s) re-conception process, or can be used to facilitate the users information search by personalization (through dynamic inserted links for example).

As we also saw in Chapter 1, so far among the agglomerative hierarchical clustering methods, only the classical AHC was been used to analyze Web data. We remind the most successful one, the incremental BIRCH technique [ZRL96], which handles very well large data collections.

In this context, our applicative study of the 2-3 AHC is carried in the Web Mining field following our research team* objectives.

The objectives of our research team, AxIS, are related to the design, analysis and improvement of the information systems (IS), driven by usage. Although in the short run the project is directed mainly towards Web sites and/or Web services, we place ourselves in a global point of view of design and evaluation of adaptive information systems based on the W3C† standards. The word "adaptive" represents both the ability to adjust to the user (personalization), and the ability to learn from usage analysis. One of our team interests is the study of the INRIA's Web site activity and structure.

Our two main applications presented in the next two chapters are following this study and present the clustering of INRIA's research teams through its Web usage and activity reports. More precisely, these applications concern:

- first in Chapter 5, a clustering of the INRIA's research teams based on their Web sites via the usage mining (i.e. Web users behaviours). This analysis is related to the WUM field;

*AxIS research project, `http://www-sop.inria.fr/axis/`

†World Wide Web Consortium: `http://w3c.org/`

- secondly in Chapter 6, a clustering of INRIA's research teams based on the 2003 XML activity reports via content and structure mining. This analysis is related to the XML Document Clustering field.

Chapter 5

Web Mining Application

In this Chapter we investigate the use of our 2-3 AHC algorithm as classification/clustering method in a Web Usage Mining (WUM) analysis. according to our knowledge, this is a first study of applying an hierarchical classification method on Web Usage data.

The presented analyses consist in clustering the visited topics of INRIA's* Web sites, using our 2-3 AHC algorithm. This was done by analyzing visitors activities on INRIA's Web sites, i.e. their navigational behaviours.

The objective is to study the Web users perception of INRIA's research activity, by using their browsing behaviour. As a direct result of this analysis, one could propose different improvements in INRIA's Web site to meet the users needs.

For these analyses we have compared the obtained classifications with INRIA's scientific organization. This scientific organization of research teams into research themes has changed in 2004, so we analyzed the Web users behaviour during two time periods: before and after the change.

Before presenting the details of our application in Section 5.4, we begin by introducing the main terms used in the WUM field in Section 5.1, followed by a short description of the WUM data (the log files) process and the WUM steps in Section 5.1.2.

Then Section 5.2 will introduce our analysis motivations. Before concluding in Section 5.5, we present our INRIA's research teams clustering and result interpretation in Section 5.4.

*The French National Institute for Research in Computer Science and Control

5.1 Introduction

We present first the main terms used in the WUM process as defined in [TT04, Tan05a], which are based on the ones used by the World Wide Web Consortium* (W3C) for the Web characterization terminology [LN99].

Then in Section 5.1.2 we will make a short description of the main source of information in a WUM analysis, that is the Web log files.

5.1.1 WUM Terms

Definition 13 : *A* **resource***, according to the W3C specification, can be "anything that has identity", and is also called an* **Uniform Resource Identifier** *(***URI***). Examples include an HTML file, an image, and a Web service.* □

Definition 14 : *A* **Web resource** *is a resource accessible through any version of the HTTP protocol (for example, HTTP 1.1 or HTTP-NG).* □

Definition 15 : *A* **Web server** *is the server that provides access to the Web resources.* □

Definition 16 : *A* **Web page** *is the set of data constituting one or several Web resources that can be identified by an URI. If the Web page consists of n resources, the first* $n-1$ *are embedded into the* n^{th} *URI, which identifies the Web page.* □

Definition 17 : *A* **page view** *(also called* **hit***) occurs at a specific moment in time, when a Web browser displays a Web page.* □

Definition 18 : *A* **Web browser** *or* **Web client** *is a client software that can send Web requests, handle the responses, and display requested URIs.* □

Definition 19 : *A* **user** *is a person using a Web browser.* □

Definition 20 : *A* **Web request** *is a request that a Web client makes for a Web resource. It can be explicit (user initiated) or implicit (Web client initiated). The explicit Web requests are also called clicks.* □

*http://www.w3c.org

Definition 21 : *A* **user session** *consists in a delimited number of an user's explicit Web requests across one or more Web servers.* □

Definition 22 : *A* **navigation** *or* **visit** *represents a subset of consecutive page views from an user session occurring close enough (measured by means of a time threshold, usually 30 minutes).* □

Definition 23 : *The* **Browsing Speed (BS)** *represents the user navigation speed and is given by the average number of seconds spend per page by the user in a navigation:*

$$BS(N) = \frac{|N_{pages}|}{N_{duration}}$$

where $|N_{pages}|$ *is the number of visited pages in the navigation* N *and* $N_{duration}$ *is the duration of the navigation.* □

5.1.2 Log Files

The main source of information in a WUM analysis are the logs files generated by the Web servers. These contain all the Web requests made by the Web site visitors over a time period and ordered chronologically.

Definition 24 : *A* **Web server log file** *contains requests made to the Web server, recorded in a chronological order.* □

The most popular log file formats are the *Common Log Format* [W3C95] (CLF) and its extended version, the *Combined Log Format*, denoted ECLF [Tea95].

A line in the ECLF log file contains information about an unique Web request. The ECLF format is the following:

[IP] [name] [login] [Date] [Type] [URI] [Status] [Size] [Referrer] [User Agent]

where:

IP	The client's host name or its IP address
name	The client inetd id (generally empty and represented by a "-")
login	The user login (if applicable)
Date	The date and time of the request
Type	The request type (GET, POST, HEAD, etc.)
URI	The requested resource name
Status	The request status
Size	The size of the requested resource
Referrer	The referrer of the request which is the URL of the Web page containing the link that the user followed to get to the current page
User Agent	A string identifying the browser and the operating system (denoted UA in the followings)

A small example of an ECLF line extracted from INRIA's Web log is presented in Figure 5.1 below. We can see that an user from the IP address 194.254.174.176, requested on the April 5^{th}, 2005 at 19:17:49 the page `/axis/Publications/show.php?keyword=KDD`. The request was of type GET using the HTTP 1.1 protocol, and also a successful one (status code 200) with the size of the transfered page of 52382 bytes. The user arrived on this page from Google via a search with the keywords "WEB USAGE MINING" and used Microsoft Internet Explorer 4.01 from his/her PocketPC device.

194.254.174.176 - - [05/Apr/2005:19:17:49 +0200] "GET /axis/Publications/show.php?keyword=KDD HTTP/1.1" 200 52382 "http://www.google.fr/pda?q=WEB+USAGE+MINING&hl=fr" "Mozilla/4.0 (compatible; MSIE 4.01; Windows CE; PPC; 240x320)"

Figure 5.1: A Web Request from INRIA's Web Server Log (ECLF Format)

5.1.3 WUM steps

The Web servers collect large volume of data in their logs from the Web site usage. As we said before this is the main source of information in a WUM analysis, but complementary available information helps to improve the analysis results. Thus, information

on the Web site structure, on the content of the Web pages, on user profiles, etc. can be used to improve the data quality.

There are three main steps in a WUM analysis [Tan05a]: data preprocessing, pattern discovery and pattern analysis (see Figure 5.2).

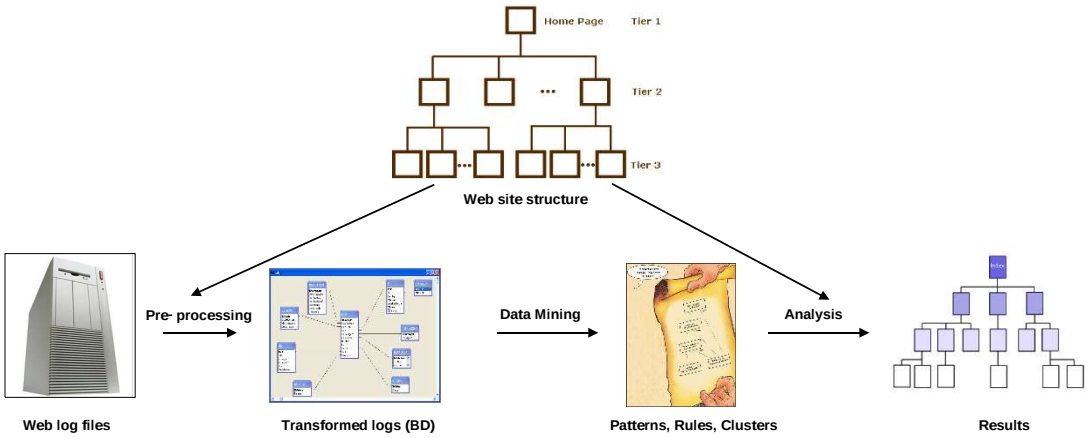


Figure 5.2: Schema of a General WUM Process

For our WUM analysis (cf. Section 5.4), we begin with a brief description of the analyzed data and the analysis motivations in Section 5.2. The data preprocessing step consists in the cleaning and preprocessing of the raw data stored in the log files and is presented in Section 5.3. Section 5.3.2.2 presents our extension of the relational DB model used in [Tan05a]. The analyses (pattern discovery) and their results interpretations (pattern analysis) are then presented in Section 5.4, followed by the discussion and perspectives in Section 5.5.

5.2 Data Description and Motivations

INRIA has six research units across France (Futurs, Lorraine, Rhône-Alpes, Rennes, Rocquencourt, Sophia Antipolis) and seven Web servers: one for each research unit plus a national one. Users searching for information on one of INRIA's Web sites, are transparently browsing through the interconnected pages of its Web servers. To trace their behaviours, one must analyze the log access files from these Web servers.

While searching for informations on INRIA's Web sites, users can visit the pages of different research teams that have common scientific objectives. A INRIA research team is a team of a limited size with relatively focused scientific objectives. Based on their scientific objectives, INRIA has grouped research teams into research themes.

On April 1^{st} 2004, INRIA's scientific organization into research teams has changed from the four existing research themes, into five new research themes (see Appendix E).

In this context, we decided to analyze the impact of INRIA's Web sites structure on users navigations before and after this change. This was done by performing two different analyses of users visits: one on INRIA's first level topics before the change and another on INRIA's research teams before and after the change. An additional analysis to compare the classical AHC and our 2-3 AHC on a single research theme was also performed.

We chose to analyze the log files from two of INRIA's Web servers: the national Web server (*http://www.inria.fr*) and also the Sophia Antipolis research unit Web server (*http://www-sop.inria.fr*).

As a secondary source of information beside the log files, we used some additional information on the both Web site structures to assign Web pages to research teams, and thus to be able to cluster the research teams based on their group of Web pages and on the Web users behaviours.

Since INRIA's scientific organization into research teams has changed, we have chosen to study the Web users behaviours on two different 15 days time periods before and after this change:

- from 01 until 15 January 2003, period denoted in the followings as Per_1,
- and from 27 May until 10 June 2004, period denoted as Per_2.

Indeed, the main motivation of our study here was to analyze the impact of the changes in the national Web site structure (see Appendix E), on users behaviours when searching for information. For the first analysis this concerned the impact of the entire Web sites on users behaviours, whilst for the second analysis we studied the impact of the national research presentation pages* .

More particularly, our study concerned the clustering of INRIA Web sites' visited topics (corresponding to actual research teams), using the 2-3 AHC and was done in two phases:

- first, the *preprocessing* of the Web access logs (based on the work from [TT04]) and presented in section 5.3,
- secondly, the data mining (using our 2-3 AHC algorithm) and the result analysis phase (cf. Section 5.4.2).

*The national Web site changes actually concerned only 20 Web pages used for presentation of the research teams (see Table E.1)

5.3 Extended Relational DB Model for Data Preprocessing

We begin this section by introducing in Section 5.3.1 the used data preprocessing methodology which was proposed by Tanasa et al. (see. [TT04, Tan05a] for more details). We used the AxIS LogMiner* tool developed within the AxIS research team at INRIA Sophia Antipolis.

Then, we will extend in Section 5.3.2 the relational DB model proposed in [Tan05a]. This model is currently available in the new version of the AxIS LogMiner tool.

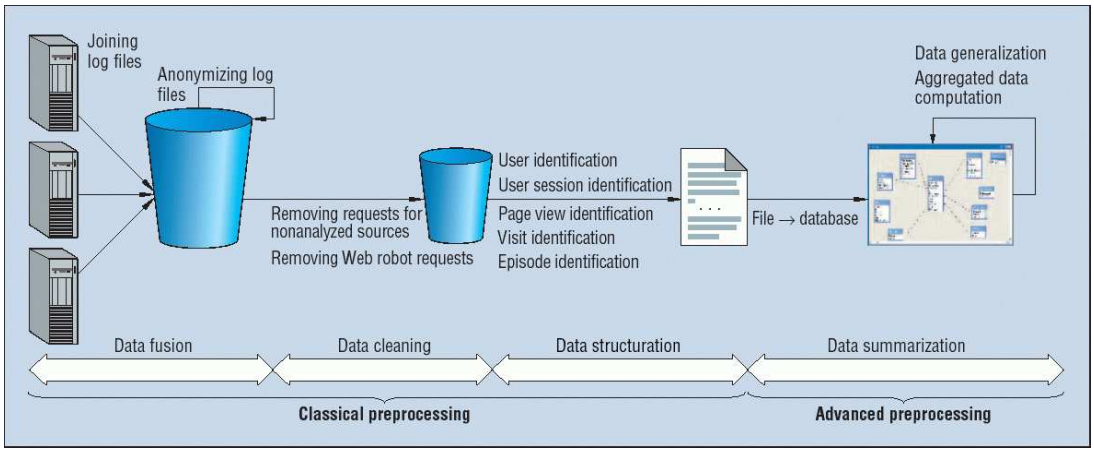


Figure 5.3: The data preprocessing steps

5.3.1 Data Preprocessing

The aim of the preprocessing phase was to identify and extract *user navigations* (cf. Definition 21) from the raw Web logs, and was done in four steps [Tan05a]: *data fusion*, *data cleaning*, *data structuration* and *data summarization*. Figure 5.3 presents these steps, further details can be found in [TTM04, Tan05a].

During the *data fusion* step, the Web logs files were joined together for each analyzed period (resulting in log L_1 for Per_1 and log L_2 for Per_2), to reconstruct the cross-server users' navigations.

Thus the two joined logs contained all the requests (chronologically sorted) made by different users for different resources on the two Web servers, over the given periods of time. To distinguish the requests made on one server from the one made on the other, we added at the beginning of the accessed resource, the server name: http://www.inria.fr or http://www-sop.inria.fr. This helps also to reconstruct the full

*Tool description available at: `http://www-sop.inria.fr/axis/axislogminer`

URI of the accessed resource, which can be then used to reconstruct users navigations [CDSL+05], by comparing it with the referrer field of the following pertinent requests (when available).

After joining the log files, an anonymization of certain informations was performed. This is necessary for privacy reasons in case that the log file or the analysis results are made public.

The following fields were anonymized:

- IP address: the real IP address or host name is replaced with an identifier that keeps the information about the domain extension, which is the country code (i.e. .fr, .de, .ro) or the organization type (i.e. .com, .edu, .org). For example *gicu.olimpia.edu* becomes *12578.example.com.edu*, where *12578* is an ID used to retrieve when needed the original real name;
- the login name when available;
- the email addresses present in the UA field, which is actually a very rare case.

Since some of these requests from the log files were made for non-relevant resources from our analysis viewpoint, these were eliminated in the *data cleaning* step. For example, we do not interest ourselves in requests for images, since they usually are implicit requests (images contained in the accessed page [Tan05a]).

But sometimes the images can be explicitly accessed by the user via a click, and constitute in this case a valid Web request which must not be eliminated from the analysis. To identify this particular case of images, the LogMiner tool uses additional information on the analyzed Web sites.

Other types of eliminated requests include the javascript files (.js), the style sheet files (.css), the flash animations (.swf), etc.

A big part of an Web site traffic is caused by the web crawlers, commonly known as Web robots. These are computer software from search engines, that periodically harvest the Web site content to update the index. Usually the number of requests made by a Web crawler is larger than the one of a normal Web user and can be also identified by their User Agent (UA) field.

To identify a Web robot the following three heuristics were used:

- if they requested '/robots.txt';
- if their UA is known (i.e. GoogleBot, Yahoo Slurp, etc.)
- if their browsing speed (see Definition 23) is superior to a certain threshold. This step must be performed after the navigations identification to be able to compute

the BS. Next, if $BS(N) > 2$ and $|N_{pages}| > 15$, the navigation N comes most probably from a Web robot and can be eliminated along with all its requests from the analyzed log file.

Filtering out all these requests has reduced L_1 to 11% and L_2 to 15%, from their original size. For example for L_2, the number of requests was reduced from 4.473.228 to 686.084, equivalent to a log file size reduction from 901Mb to 135Mb.

Next, the *data structuration* step groups the unstructured log files requests by *user*, *user session*, *page view*, and *navigations*.

Today, there is no perfect solution to identify a Web site users from the log file due to the poor information within these files, due to the proxy servers, dynamic addresses (DHCP), personal security software*, and cases where multiple users use the same computer or one user uses multiple browsers and/or computers. Several techniques can provide additional information on the users re-visiting a Web site: user registration, cookies or modified browsers.

In the case of the ECLF log files the current best solution is to use the triplet (*IP*, *Login*, *UA*) [Tan05a]. In most cases (ours included), this is not the best solution since the *Login* field corresponds for example to the username used via the .htaccess/.htpasswd authentification method in Apache Web server. Since these are directory specific, the same user can login in different parts of the same Web site using different usernames. For example an user might use an username to access the intranet of its research team, and another username to access a working group intranet found on the same Web site.

Therefore, we consider as a *user*, the couple (IP, [User Agent]) in our analysis, and even though this can be sometimes inaccurate, it still has an 92.02% precision [Tan05a] in identifying Web users.

The *user session* represents all the actions performed by the user over the analyzed period. There are 115.825 identified sessions for Per_1 and 96.984 for Per_2.

Users *navigations* are obtained by splitting every user session using a 30 minutes threshold. This is done when two successive requests of the same user are separated by a time period superior to the given threshold. For our analysis we obtained 173.015 navigations for Per_1 and 145.454 navigations for Per_2.

Finally, the obtained log file is stored in a relational database in the *data summarization* step presented in the next Section 5.3.2.

5.3.2 Extension of the Relational DB Model

The log file obtained so far is stored in a database using the relational model from [TT04, Tan05a] and presented in Figure 5.4. We extended this model to be used by

*Some personal security software are blocking the request informations: referrer, cookies, etc., which makes the identification task almost impossible

our team, by attaching new tables or new attributes to the existing ones.

5.3.2.1 Extended Model

The main table of the model is the LOG table that contains information about all the requests. Each row in the table corresponds to a Web request and contains mainly only IDs (foreign keys) linking to the other tables in the model.

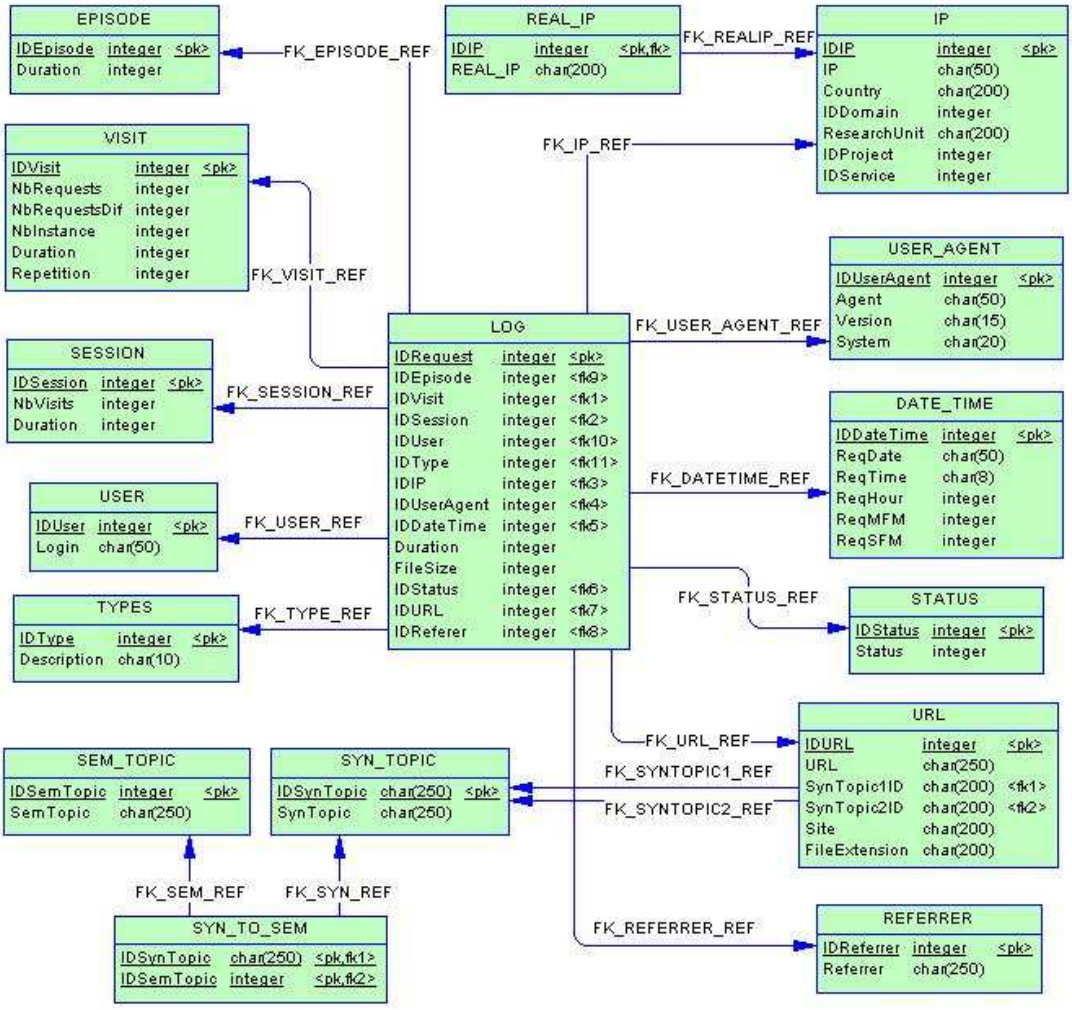


Figure 5.4: DB Relational Model for Log Files from [Tan05a]

After storing the existing information in the DB, a *generalization* of the URLs is performed to extract the first and the second syntactic topics (see Figure 5.4 and Figure 5.6). These syntactic topics were then mapped into semantic topics like: *projects*, *people*, *services*, etc. These semantic topics were manually generated [TT04].

Only the first two syntactic topics are directly stored in the URL table, while the others are discarded. But the greater level (3^{th} and on) syntactic topics can also contain important information on the analyzed URL.

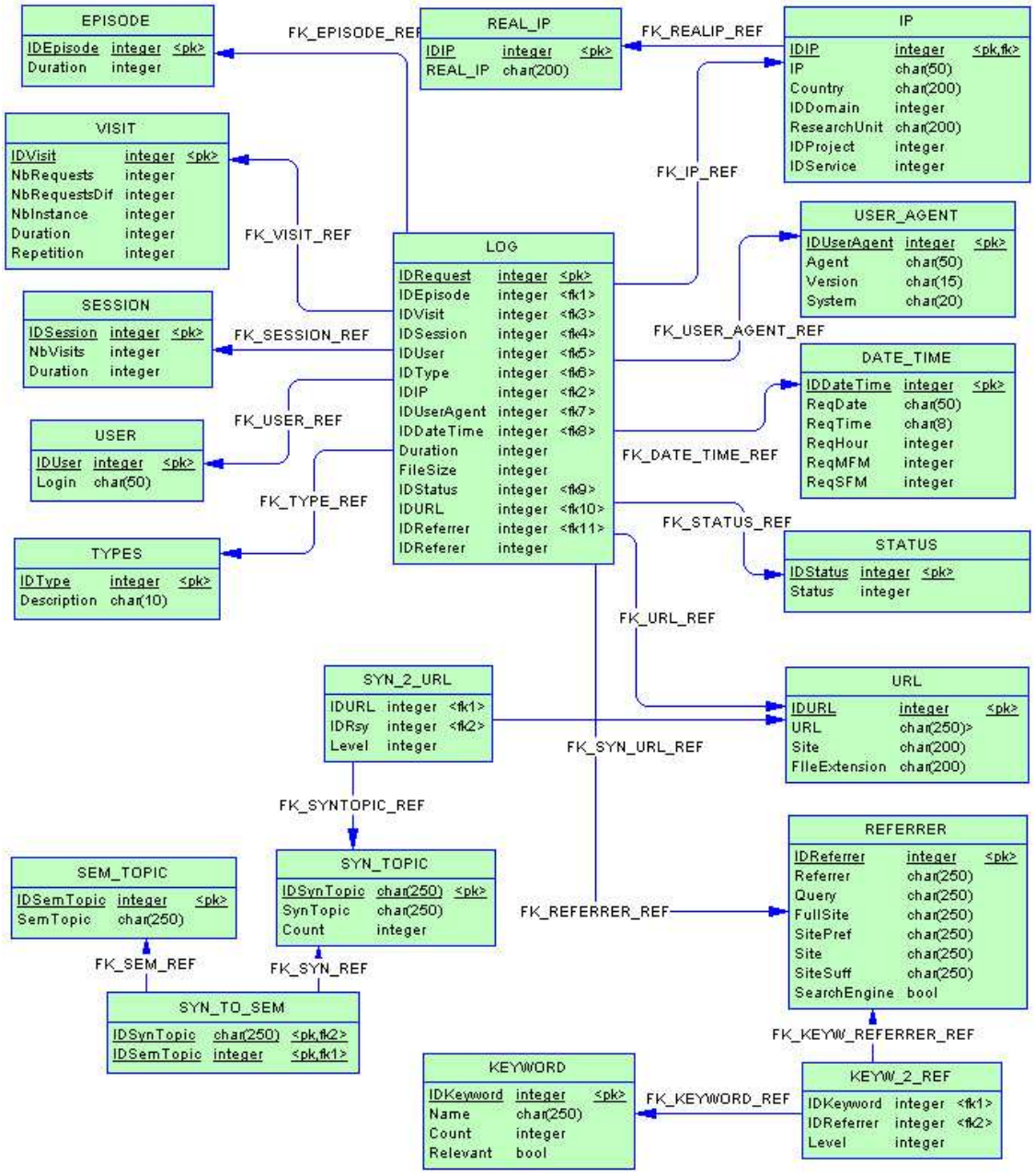


Figure 5.5: Extended DB Relational Model for Log Files

$$http://\underbrace{www-sop.inria.fr}_{site}/\underbrace{axis}_{topic1}/\underbrace{personnel}_{topic2}/\underbrace{Doru.Tanasa}_{topic3}/doru-eng.html$$

Figure 5.6: URL Generalization

Therefore, we extended the model to store all levels syntactic topics. For this, we modified table SYN_TOPIC and we added the table SYN_2_URL (cf. Figure 5.5). In the second table the links between all syntactic topics of each URL and their URL are stored, not only the first two levels. This will be later used to assign URL to research teams (see Section 5.4.2).

Topic	**Hit Count**
rapportsactivite	*26019*
personnel	10808
RA2001	*9839*
cgi-bin	5988
rrrt	4319
RA99	*3145*
RA2000	*2819*
epidaure	2693
RA95	*2641*
robotvis	2591
RA96	*2561*
semir	2514
RA98	*2440*

Table 5.1: Most Visited Topics on INRIA's Web Sites on Per_1

Using this extended model to generalize URLs, we found that among the most visited topics by Web users, the topics from the activity reports of the research teams (*rapportsactivite*, *RAxx* or *RAxxxx*) were clearly the most visited ones.

For example in Table 5.1 for the first analyzed period, Per_1, the *rapportsactivite*

topics had 26019 hits, followed by *personnel* with 10808 hits. Moreover, seven out of the first 13 most visited topics, belong also to the activity reports topics and all topics related to activity reports accounted for 22.77% of the total hits (53618 out of 235473) for this period.

Due to the great interest of the Web users in INRIA's activity reports (RAs), we studied in Chapter 6 their structure and content. For this we analyzed the collection of XML documents representing the activity reports of INRIA's research teams from 2003.

Using this extended model for the URL analysis, we obtained 6.383 syntactic topics for Per_1 and 6.542 for Per_2 (see Table 5.3) compared to 348 for Per_1 and 568 for Per_2 using the previous model (cf. Figure 5.4).

For other analyses planned on same logs (by IUT* students and other AxIS members), we also modified the REFERRER table and added another two tables, KEYWORDS and KEYW_2_REF (see Figure 5.5). More information on the origin of the visit and its characteristics can be thus stored. Indeed, the referrer field contains, when present, important information on the origin of the request (see Figure 5.1) which can be very useful in a WUM analysis. This information can be interpreted based on the URL type: internal page (page form the analyzed Web sites) or external page (external link). In the first case, when an user is navigating through the analyzed Web sites, the referrer represents the previous visited pages and can help us better reconstruct the user's cross-servers navigation.

But sometimes, the user arrives on the analyzed Web sites from a search engine or an external link (usually in the beginning of the navigation). This is a very rich source of information especially for e-commerce Web sites where one can identify the source of the visitors that purchased from their Web site: search engines, payed directory listings, e-mail campaigns, etc. Moreover, the keywords used by users in the external search engines can be extracted and analyzed to eventually improve the site ranking or to see which keywords produce the most benefit/sales, etc. Also additional information on the country of origin, the user's behaviour, etc. can be extracted and used in different analyses.

In our analysis on INRIA's Web site, one can use this type of data to determine for example the research teams similarity based on the keywords used in external search engines [TMT06] (see Section 5.3.2.2). The same data generalization can be applied for internal search engines, i.e. search engines within the analyzed Web sites.

Extracting keywords from the referrer is not an easy task, since each search engine use its own keywords representation in the URL. But fortunately, the most common

*`http://stid.unice.fr`http://stid.unice.frff

way of representing the search keywords in the URL, is using the q parameter like the example from Figure 5.1. In [TMT06], from a total of 649.328 detected parameters in the referrer, 127.413 where search parameters (like *q=keywords*) from which 117.966 (92.59%) where represented by the q parameter. To detect the rest of the search parameters, we manually defined some extraction rules. For example, if the search engine is Yahoo and the parameter is p, the values of p represent search keywords, while for other search engines p has another purpose.

The extended relational DB model that we proposed is now available in the new version of the AxIS LogMiner* tool.

5.3.2.2 Other Applications of the Extended DB Model

The extended DB relational model that we proposed in Section 5.3.2, was used in [TMT06, LMTT06] to generalize Web pages based on the keywords used by Web users on search engines. This represents information about the users' access to these pages, using the referrer data. The method was called Generalized Web Usage Mining or GWUM.

The keywords are then processed with the TreeTager tool developed by H. Schmidt [Sch94] at Stuttgart University, which lemmatizes the words of a text and labels them with grammatical annotations. The authors used the keywords frequently employed in search engines to access INRIA's Web pages. The keywords were then used to describe these pages. But because of the large number of Web pages (62.721) and obtained keywords (35.367), a direct construction of a dissimilarity matrix was impossible.

Therefore, the keywords are first clustered using a dissimilarity matrix based on the Jaccard index [Sne57], as in our previous analyses (cf. Section 3.5). Then our HCT toolbox (see Section 4.2) is used to apply our 2-3 AHC algorithm (V3 with integrated refinement) on the keywords to extract keywords "categories".

The obtained keywords clusters were next used to assign each page to the cluster containing the keywords that describe it best. Finally, the URL (Web pages) were replaced in the original log file with their categories (keyword clusters identifiers) and sequential patterns were extracted from the log file composed of original and generalized requests. Only pages accesed from an external search engine (having search keywords in referrer) where generalized.

The results, showed that using the generalized Web pages, generalized patterns with higher support and easier to interpret can be obtained compared with a classical sequential pattern analysis.

*`http://www-sop.inria.fr/axis/axislogminer`

5.4 Research Teams Clustering Based on Web Users Behaviours

After storing the data in the relational DB using the extended model, a two step *advanced data preprocessing* was performed (see Section 5.4.1) for our analyses to select a relevant dataset for matrix generations. The analyses and their results interpretations are presented then in Section 5.4.2.

5.4.1 Advanced Data Preprocessing

First, we performed a general *data selection* step in which we select from the relational DB the navigations (users visits) that where used in our analyses. The purpose of this first data selection is to select the user navigations to analyze.

We used the following three criteria to obtain "pertinent" user navigations to analyze:

- navigation duration $N_{duration} > 60$ seconds,
- number of requests in the navigation $N_{pages} > 10$,
- browsing speed $(N_{pages}/N_{duration})$ $BS(N) \leq \frac{1}{4}$.

This has reduced the number of analyzed navigations to 9625 for Per_1 and to 9309 for Per_2 (see Table 5.2 below).

Periods / Data	Per_1	Per_2
Raw log lines	6.040.290	4.473.228
Selected log lines	634.811	686.084
Sessions	115.825	96.984
Total Nav.	173.015	145.454
Selected Nav.	9.625	9.309

Table 5.2: Data preprocessing

Secondly, depending on the analysis, we performed secondary data selections. For example, in the secondary data selection associated with our first analysis (section 5.4.2.1) we decided to keep only the visits on both INRIA's servers and to cluster the visited first level topics (from the visited URLs). Thus the number of analyzed

navigations was reduced to 3905 for Per_1 and to 3513 for Per_2. Also, the Web pages returned by the Web server with an error status code (≥ 400) were ignored in our analysis.

Since INRIA research teams organization has changed (starting from 1^{st} of April 2004), its Web site structure changed accordingly. The research teams were reorganized from the four existing research themes, into five new research themes (see Appendix E).

We decided to analyze the impact of the Web site structure on users navigations before and after this change (during Per_1 and Per_2). This was done by performing two different analyses of users visits: one on INRIA's first level topics and another on INRIA's research teams. For the second analysis, since an user visit is actually a set of visited URLs, we needed to determine which URLs belong to different research teams.

As we said before, a *data generalization* step in which the visited URLs were assigned to different research teams for later clustering, was performed. From the total of number of visited topics (see Table 5.3 below), after the data selection step we obtained 190 visited topics for Per_1 (78 were research teams). For Per_2 we found 210 topics from which 86 were research teams (49 actual research teams and 37 old research teams from Per_1).

Periods / Data	Per_1	Per_2
Raw log lines	6.040.290	4.473.228
Selected log lines	634.811	686.084
Sessions	115.825	96.984
Total Nav.	173.015	145.454
Selected Nav.	9.625	9.309
Both Servers Nav.	3.905	3.513
Total Topics	6.383	6.542
Selected Topics	190	210
Teams	78	86

Table 5.3: Summary of data preprocessing

Each URL can have several topics associated with different semantic topics. In our analyses we interest ourselves in the *project* semantical topic, which includes the

research teams only.

In a first step, a URL was assigned to a research team when one of its topics was the research team itself. After this, complementary information on INRIA's Web site was used to assign URLs to research teams. For example, the URL:
`http://www.inria.fr/recherche/equipes/axis.en.html` does not contain any research team topics, but is the AxIS research team presentation page from INRIA's main server.

The data preprocessing performed so far on the two time periods is summarized in Table 5.3.

Dissimilarity Matrix Generation:

After the data generalization step and in order to cluster the obtained topics, we needed to *compute the dissimilarity matrix* used as input for the AHC and our 2-3 AHC algorithm*. For this, we used the Jaccard similarity index on the visited topics as defined in [GCGR+04]. We represented each navigation by a binary vector of the visited topics: the position i in the vector is 0 if topic T_i was not visited and 1 if topic T_i was visited during the navigation N_i (see Table 5.4).

Navigations / Topics	N_1	N_2	$\cdots$	N_{3905}
T_1	1	1	$\cdots$	0
T_2	0	1	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T_{190}	0	0	$\cdots$	0

Table 5.4: Binary table for Per_1 describing the navigations using the visited topics

Based on these vectors and aiming to define a similarity/dissimilarity between two topics T_i and T_j, we define the four following quantities (see also Table 5.5):
- a as the number of counts when $T_i^k = T_j^k = 1$,
- b as the number of counts when $T_i^k = 0$ and $T_j^k = 1$,
- c as the number of counts when $T_i^k = 1$ and $T_j^k = 0$,
- d as the number of counts when $T_i^k = 0$ and $T_j^k = 0$.

*The 2-3 AHC algorithm with refinement (V2) was used in this case, the blind merging avoidance (V3) not beeing yet implemented

T_j \ T_i	***0***	***1***
0	a	b
1	c	d

Table 5.5: Quantities used for topics similarity computation

Then the similarity between two topics T_i and T_j is computed using the Jaccard coefficient [Sne57]:

$$S(T_i, T_j) = \frac{a}{a+b+c} \tag{5.1}$$

The obtained similarity represents the probability of visiting both topics when at least one of them is visited. The dissimilarity matrix used as input for the classical AHC and our 2-3 AHC, was computed using the dissimilarity: $\mu(T_i, T_j) = 1 - S(T_i, T_j)$

5.4.2 Analyses and Results

We present below the three analyses that we performed and their results. The first two ones concern the chosen field, the third one concerns the comparison of 2-3 AHC and classical AHC.

5.4.2.1 Impact of the Global Web Site Structure on Research Teams Clustering (Per_1)

For our first analysis, we have focused on the research teams distribution in the server-crossed visited topics. The main motivation for this analysis was to study the impact of the entire Web sites on user navigations before the research teams reorganization. For this we compared the teams clustering obtained with our 2-3 AHC algorithm with their existing organization into research themes. This analysis is similar to the one done by other members of the AxIS team and presented in [GCGR$^+$04], to which we compare our results.

For this analysis and to compare ourselves to [GCGR$^+$04], we have selected from Per_1 the server-crossed navigations, that are visiting both Web sites: main and Sophia's (3905 navigations). Then based on these navigations we constructed a dissimilarity matrix on the first level visited topics, that we clustered using our 2-3 AHC algorithm.

Table 5.6 presents the repartition of the research team topics in the obtained clustering (the other topics are not presented here). Also, we did not represent the one el-

ement clusters (the "outliers") that were obtained: `caiman 4B, saga 2B, meije SOP 1C, sysdys SOP 4B, chir 4A, cafe 2B, codes 2B, visa SOP 4A, tropics 1A, omega 4B`.

We added after the name of each research team, their theme and sub-theme, as well as their site (empty for the main site and "SOP" for Sophia's site).

As we can see the research teams distribution usually corresponds to their theme membership: 16 out of the 19 non-trivial clusters (84%) contain research teams from the same theme.

We also noticed that old research teams that have been replaced by new research teams, are in the same clusters as the corresponding new ones. This is due to the fact that their pages are strongly interconnected. For example: *aid* was replaced by *axis* (cluster 7), *rodeo* by *planete* (cluster 13), etc.

The same analysis was performed in [GCGR+04]. The authors used the Per_1 log files form both INRIA's Web servers, the preprocessing methodology from [TT03] and the DB relational model from Figure 5.4.

The first level visited topics where analyzed and clustered. After constructing the dissimilarity matrix on these topics with the same methodology described in Section 5.4.1, the topics where clustered using an adapted version of the Batch SOM* method [Koh01].

Using this method, the authors obtained 12 clusters containing the visited topics. Only 6 of these clusters (50%) contained research teams from the same research theme, which can be explained by the exclusion from the analysis of the higher level topics (second, third, etc.).

*Self Organizing Maps

robotvis SOP **3B**, robotvis **3B**, epidaure SOP **3B**, odyssee SOP **3B**, epidaure **3B**, ariana SOP **3B**, ariana **3B**	comore SOP **4A**, icare SOP **4A**, icare **4A**, miaou SOP **4A**, reves SOP **3B**, miaou **4A**, chir SOP **4A**, comore **4A**, caiman SOP **4B**	orion SOP **3A**, axis SOP **3A**, orion **3A**	prisme SOP **2B**, prisme **2B**
koala SOP **2A**, koala **2A**, croap SOP **2A**, croap **2A**	odyssee **3B**, dream SOP **3A**, lemme **2A**, opale SOP **4B**, opale **4B**, certilab **2A**, pastis **3B**	orion SOP **3A**, acacia SOP **3A**, acacia **3A**, axis SOP **3A**, orion **3A**, aid SOP **3A**, aid **3A**	coprin SOP **2B**, saga SOP **2B**, saga **2B**
sinus SOP **4B**, sinus **4B**, smash SOP **4B**	robotvis SOP **3B**, robotvis **3B**, odyssee SOP **3B**	mimosa SOP **1C**, mimosa **1C**, tick SOP **1C**, tick **1C**	sloop SOP **1A**, sloop **1A**, oasis SOP **2A**, oasis **2A**
rodeo SOP **1B**, rodeo **1B**, planete SOP **1B**, planete **1B**	lemme SOP **2A**, tropics SOP **1A**, mascotte SOP **1B**, omega SOP **4B**, galaad SOP **2B**, cafe SOP **2B**, certilab SOP **2A**	mistral SOP **1B**, mistral **1B**	mefisto SOP **4B**, mefisto **4B**
mascotte SOP **1B**, mascotte **1B**	safir SOP **2B**, safir **2B**	meije SOP **1C**, meije **1C**	

Table 5.6: INRIA's Web site topics clustering using 2-3 AHC for Per_1

5.4.2.2 Theme 3 Analysis on Both Time Periods

The purpose here was to analyze the influence of the INRIA's main Web site structure (and its implicit research themes organization) on the visited topics of the Web users for research theme 3 during both time periods. Since theme Cog regrouped most of the old theme 3 research teams, we also analyzed it for the second period.

For this analysis we have selected from the obtained navigations in section 5.4.1, only those visiting at least one page on INRIA's main server. From these navigations, we choose to analyze the visited topics (research teams) only from the main server pages and to cluster only the research teams topics for theme 3 (Per_1 and Per_2) and for theme Cog (Per_2).

Analysis 2a:

First we have selected only those navigations containing at least one visit of the theme 3 pages on INRIA's main server during Per_1. From these navigations, we have clustered the visited topics on the main server only, corresponding to research teams from theme 3 (Figure 5.7).

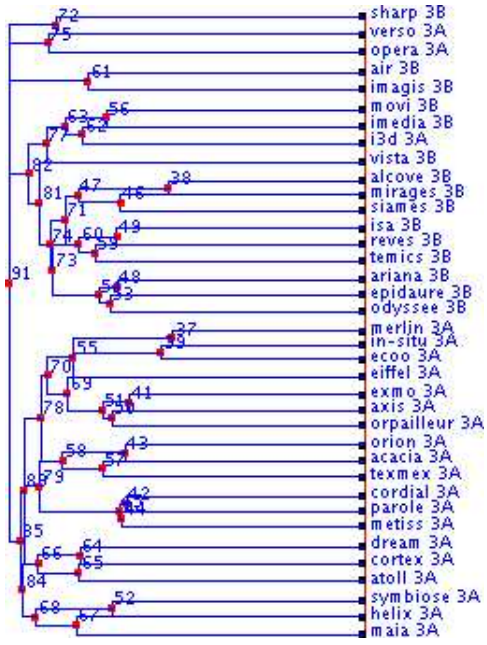


Figure 5.7: 2-3 Hierarchy on theme 3 projects during Per_1

In the resulting classification we can distinguish two main clusters (82 and 85) that group almost all elements from the two sub-themes of theme 3 (see Appendix E). This confirms that users are usually visiting teams from the same research subtheme for

the theme 3 during the first time period. Same behaviours were obtain for the other research themes.

Analysis 2b:

Next, we bassically performed the same analysis as the previous one (2a) but on the second time period. Since one can find teams from the old theme 3 in each new theme, we have selected for Per_2 the navigations visiting at least one of the new themes pages on the INRIA's main server. From these navigations we have focused again on the topics corresponding to research teams from the "old" theme 3, only on the main server's pages (cf. Figure 5.8). Table E.2 in Appendix E presents the teams from Theme 3 and their new themes from the second period Per_2.

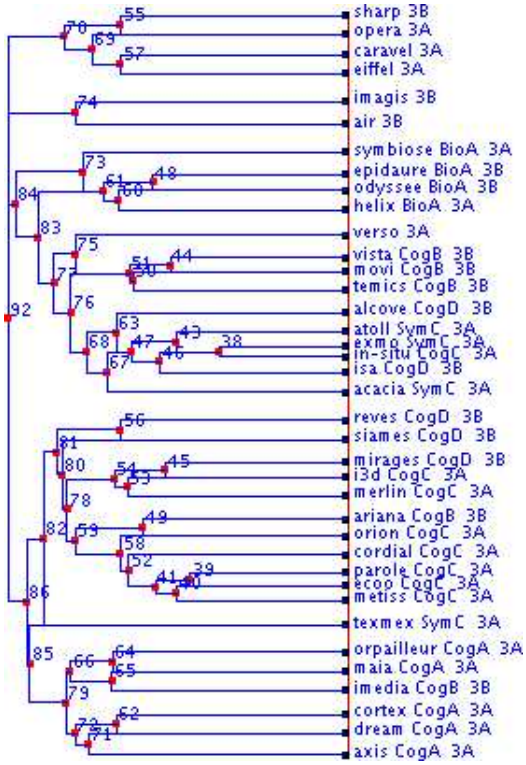


Figure 5.8: 2-3 hierarchy on theme 3 projects during Per_2

As we can see the old two subthemes 3A and 3B are no longer clearly separated, but mixed together in the resulting structure for the second time period after the change. However, we discover in this classification the new theme Bio, which groups together in cluster 73 the four research teams that were separated in the previous organization (Figure 5.7).

Some of the research teams have not been assigned to one of the new five themes

since they were either replaced or stopped, but their Web pages are still accessible on the Internet (i.e. *sharp*, *opera*, *verso*).

Analysis 2c:

Finally, since theme Cog regrouped most of the old theme 3 research teams, we also analyzed it for the second period. The purpose here was to compare the obtained classification on Per_2 with INRIA's new organization of teams in themes and more especially in subthemes just as in the first analysis. Thus, we have selected only the navigations that had at least one visit of the new theme Cog pages during Per_2. We have then clustered only the topics on the main servers pages corresponding to research teams from theme Cog (cf. Figure 5.9).

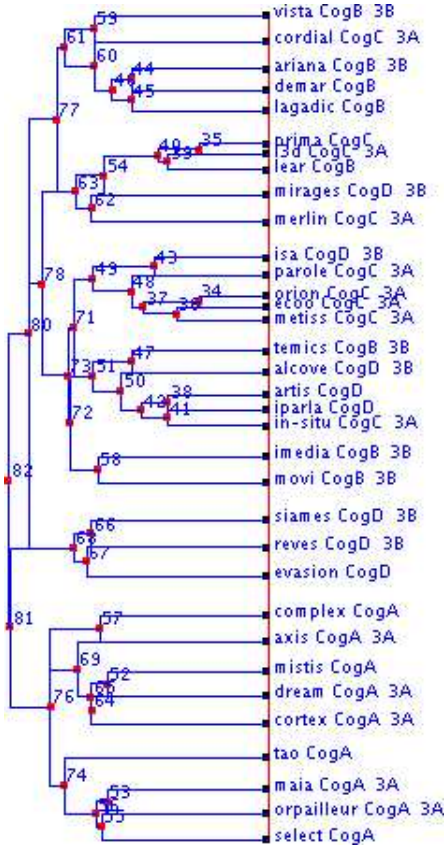


Figure 5.9: 2-3 hierarchy on theme Cog projects during Per_2

We note that users have the tendency to visit all CogA research teams, while for the others sub-themes there is a certain variability. A deeper analysis is needed to study this variability (possible causes: events like conferences or seminaries presented on INRIA's Web site that affects users visits, the recent change of structure, etc.).

To conclude, in our analyses we found that the same research teams are grouped differently on the two time periods (for example our team, *AxIS*). In order to better study the impact of the Web site structure on the Web users behaviour, we should further use the extended DB model that we proposed, to take into account different origins of the visit: search engines or national presentation pages.

5.4.2.3 Comparaison of the classical AHC and 2-3 AHC on Theme 3

In our final analysis we have compared the classical AHC method and our 2-3 AHC algorithm, by clustering the research teams from theme 3 (during Per_1). The data selection was the same as in the previous analysis: navigations visiting at least one of the theme 3 pages, topics only from the main server's pages and representing research teams from theme 3.

Figures 5.10 and 5.11 present a partial output (containing all 3B research teams) of the classical AHC respectively of our 2-3 AHC algorithm. The 2-3 hierarchy obtained contains more created clusters than the classical hierarchy (22 against 15), and thus more information. For example, analyzing cluster 54 in Figure 5.11, we can say that research teams *ariana*, *epidaure* and *odyssee* have a "stronger" probability of being visited together, compared with the one given by the classical hierarchy (Figure 5.10).

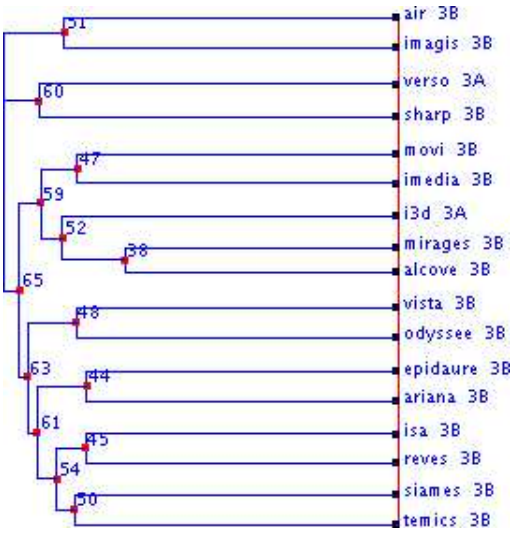


Figure 5.10: Classical hierarchy on theme 3 projects in Per_1

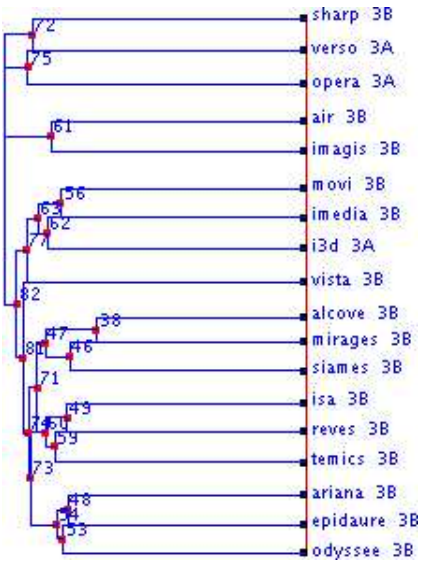


Figure 5.11: 2-3 hierarchy on theme 3 projects in Per_1

5.5 Discussion and Perspectives

This chapter presents the first application of a 2-3 AHC algorithm on Web usage data, and shows the potentials of our algorithm compared with the classical AHC algorithm.

We interested ourselves here in clustering the visited topics of INRIA's Web sites (including the research teams). We used the log files from two of INRIA's Web servers: the national one and the one located in Sophia-Antipolis. After a data preprocessing step in which all the non-relevant requests where eliminated (e.g. images, javascripts), the users and their navigations were identified and stored in a relational DB [Tan05a].

For different AxIS purposes, we improved the existing relational DB model [Tan05a] for a better generalization of the URLs and for the extraction and analysis of the search keywords present in the referrer field. Our search engine keywords extraction methodology and the proposed extended relational DB model were used in a Generalized WUM process [TMT06, LMTT06] to improve the classical WUM process (the sequential pattern mining in this case). Our extended relational DB model is currently available in the AxIS LogMiner tool* developed in our team.

For our analyses, different data selection steps were executed to generate the dissimilarity matrices between the visited topics based on users navigations and using the Jaccard coefficient [Sne57].

Since INRIA's research teams organization into research themes has changed re-

*http://www-sop.inria.fr/axis/axislogminer

cently (1^{st} April 2004), its Web sites structures changed accordingly.

In this context, we have studied the impact of INRIA's Web site structure on users navigations, during two time periods (before and right after the site structure change).

For the first period, before the change, the visited teams were correctly separated into themes and subthemes by our 2-3 AHC algorithm. For the second period, which was right after the change, we noticed that there is a certain variability in the teams visited by the Web users. A deeper analysis is needed to study this variability (possible causes: events like conferences or seminaries presented on INRIA's Web site that affects users visits, the recent change of structure, etc.). Indeed, it is important to take into account the origin of an user visit by analyzing the referrer and, when present, the search engines keywords like in [TMT06].

When analyzing the 2-3 AHC results, we found that the 2-3 AHC produced interesting results, richer than the classical AHC and better than the ones obtained in [GCGR$^+$04].

Future research directions concerning the 2-3 AHC application in the WUM field, may include the following topics:

- More detailed analyses using the referer and the status code of the accessed Web pages,
- Study of other dissimilarity measures. For example the generalized Jaccard index, which takes into account the number of visited pages for a topic, not just its presence (count vs. binary),
- Use of additional data inferred from the activities reports of INRIA's research teams (see Chapter 6), and comparison of the results with the ones obtained here.

Chapter 6

XML Document Clustering Application

In the previous chapter we analyzed INRIA's*, visited Web sites topics using the users navigational behaviours. For this, we improved the URL generalization (see Section 5.3.2) by extending the relational DB model proposed in [Tan05a]. This allowed us to extract all visited topics from two INRIA's Web sites: national and Sophia Antipolis. We thus found that the most visited topics by Web users are related to INRIA's activity reports which represent 22.77% of the total visited topics (see also Table 5.1).

In this chapter we cluster INRIA's research teams by analyzing their 2003 activity reports. These consist in a collection of semi-structured XML documents that are also accessible on the Internet. A first objective of this chapter is to study the impact on the clustering tasks of selecting different parts (sub-structures) of XML documents and different distance measures and to compare them with INRIA's research organization. The second objective is to compare the 2-3 AHC algorithms gains compared with the classical AHC (see also Section 3.5).

In Section 6.1 we will make a short introduction in the XML Document Mining area, followed in Section 6.2.1 by a description of the format of INRIA's XML activity reports.

6.1 Introduction

Document mining deals with extraction of structured information from rough text documents, as well as with the documents clustering and classification. The goal is to

*The French National Institute for Research in Computer Science and Control. `http://www.inria.fr`

improve relevant documents retrieval or to synthesise information contained in these documents, information that otherwise would be hard to extract.

There are two fundamental stages in a document mining analysis: the transformation of documents from linear strings of words into suitable data structures (text processing) and the algorithmic grouping of these representations.

Traditional approaches in document classification and clustering rely on various statistical models. A common form of text processing in many information retrieval systems is based on the analysis of word occurrences across the document collection. For this, documents representation is mostly based on a vector space or bag of words model. The number of used words/terms defines the dimension of a vector space in which the analysis is carried out. The large size of this "vocabulary" (denoted also as feature space) is usually an inconvenient for the document mining analysis. Reduction of the dimension may lead to significant savings of computer resources and processing time but if is a poor feature selection is performed, the quality of the information retrieval process can be seriously reduced.

Automatic feature selection methods have been proposed to reduce the dimension of the space. They usually try to identify representative words that can discriminate documents between various classes [YP97].

Lately, the XML documents have become a standard in information storage and transmision because of their rich and flexible format that can be easily interpreted and used by a variety of applications.

To cluster XML documents, there are two major approaches that use only their textual parts or their structure. In the first approach, the XML documents are merely reduced to their textual content which is analyzed as a normal text document. This does not take advantage of the structure of XML documents that also carries important information. In the second approach, structural similarities between XML documents are based on tree/graph-matching algorithms and are used to resemble or differentiate the XML documents.

We decided to study the impact of selecting (different) parts of XML documents for a specific clustering task. The idea is that different parts of XML documents correspond to different dimensions of the documents collection that may play different roles in the classification task. Our goal is to compare the obtained classifications with INRIA's research organization.

Two levels of feature selection are considered:

1. selection at the structure level

2. fine linguistic selection of words within the text of elements.

In the next Section we will present the analyzed collection of XML documents and our study motivations.

6.2 INRIA's Activity Reports Analysis

Every year, each research team from INRIA publishes an activity report (RA) that is made available both to the industry and to the scientific community.

Since 1994, the reports are published on the Web in HTML, PDF and PS formats. An XML format is also available since 2003 and can be used to generate the activity reports in the other formats. Next Section 6.2.1 presents the analyzed XML collection of INRIA's activity reports and the motivations of our analysis.

6.2.1 Data Description and Motivations

As we saw in the previous chapter, the most visited topics on INRIA's web sites were related to the activity reports. In this context, we chose to analyze the XML documents collection of INRIA's 2003 activity reports. The collection comprises 139 XML files, with a total of 229000 lines and 14.8 MB of data.

The structure of the XML activity reports is given by a DTD. Its top level part is given below:

```
<!ELEMENT raweb (header, moreinfo?, members, presentation,
                 foundation?, domain?, software?,  results,
                 contracts?, international?, dissemination?,
                 biblio)>
<!ATTLIST raweb year CDATA #IMPLIED >
```

Among the required sections of the documents we can find the list of team members, the objectives *presentation*, the new *results*, and the list of publications for the year (bibliography). Optional sections include research *foundation*, application domains, software, as well as international and national cooperations. Although the structure of the activity reports is predefined, the overall style and content are very flexible and unconstrained. Moreover, the teams can specify keywords for some sections to summarize their content. In the rest of this text we will call them *attached keywords* whilst the content of a section or the document will be called the *full text*.

These documents are called *semi-structured* since they present a structure, but this one is not that rigid, regular or complete like the one present in the structures of the traditional database storage systems.

As we mentioned in the previous chapter, INRIA's research teams are organized into *research themes* based on their scientific objectives. In 2003 the research teams were organized into four research themes (see Appendix E).

In this context, our main motivation was to compare this grouping of research teams into research themes with the ones obtained using the hierarchical classification algorithms. To cluster the research teams, we used their self-description containd in differents sections of the activitity report. The clustering is based on the fact that the activity reports reflect the research domains the teams are interested in and that some parts or the reports are more representative than others in identifying their research topics. For example, conferences and journals where the teams participate (publications, organization, etc.) should be an indicative of their research interests.

6.2.2 Data Preprocessing

We reused here the XML documents data preprocessing done by Despeyroux et al. [DLTV05] that we introduce briefly.

In the first data preprocessing step, all the different text elements (words) that may be relevant for the classification task, were extracted by sections from the XML documents. The extraction uses the tools described in [Des04]. After this step, each type of section is represented by the previously extracted words.

Since different sections of the activity report would play different roles in classifying the research teams, we ran five experiments using different sections of the activity reports as in [DLTV05]. This process is called "structured feature selection" by the authors in [DLTV05]. The goal of these experiments is to evaluate which parts are more relevant for the clustering task.

1. Experiment **K-F**: Keywords attached to the *foundation* part.
2. Experiment **K-all**: All the keywords attached to any of the sections.
3. Experiment **T-P**: Full text of the *presentation* part.
4. Experiment **T-PF**: Full text of the *presentation* and *foundation* parts.
5. Experiment **T-C**: Names of conferences, workshops, congress, etc. from the *biblio* section.

For these experiments, the number of extracted words in this first preprocessing step is displayed in the first column of Table 6.1.

The second processing step consists in the automatic selection of significant words within the previously extracted ones. This is known as *textual feature extraction* or

textual feature selection. Classical methods of textual feature extraction are based on statistical approaches such as the word frequency (DF) or the information gain (IG). These methods works well for large collections of texts and involve pre-processing of the full collection. For our document collection, the frequency of words may vary depending on the selected sections and the resulting collection can be very heterogeneous from one experiment to the other. To avoid heterogeneous frequency, we chose a natural language approach in which words are tagged and selected according to their syntactic role in the sentences. We use TreeTagger, a tool for annotating text with part-of-speech and lemma information, developed at the Institute for Computational Linguistics of the University of Stuttgart [Sch94].

Depending on the experiment, we select different types of words. For K-F and K-all experiments (keywords) we keep nouns, verbs, adjectives (excluding conjunctions, unknown words, etc.), whilst for T-PF and T-P experiments (full text) we keep only the nouns to limit the number of selected words. Column two in Table 6.1 displays the selected words for each experiment.

The main problem here concerns the last experiment T-C, the conference names. This is due to the heterogeneous notations used for the same conference: full name, acronyms with different format. For example the following notations are equivalent: Extraction et Gestion des Connaissances 2006, EGC 2006, EGC'06, EGC06, etc. Beside the fact that these different notations are used by different teams, they are also present among the publications of an individual team.

Therefore a normalized list of all the conference names was built manually (usually the full name) by Despeyroux et al [DLTV05] and was used to replace the other notations. Moreover, since conference acronyms are significant but unknown to the tagger, we decided not to use the tagger for this experiment, keeping all the words except the stop words (e.g. proceedings, conference).

In the last step of the preprocessing, we eliminate all the words that are not used at least by two teams. Column three of Table 6.1 contains the remaining words that were grouped in vocabularies for each experiment, while the last column represents the number of projects that contained at least one vocabulary word.

Experience	Extracted words *First step*	Selected words *Second step*	Vocabulary *Third step*	Teams
K-F	2234	1053	134	80
K-all	8671	6171	382	121
T-P	63711	16036	365	138
T-PF	320501	87416	805	139
T-C	10806	7915	659	131

Table 6.1: Size of the data for the five experiments

6.2.3 Dissimilarity Matrix Generation

For the clustering of the research teams, we use the previously obtained vocabularies (for each experiment).

For each experiment, the vocabulary W has m words, $W = \{w_j,\ j = \overline{1,m}\}$, whilst the set of analyzed teams T has n documents, $T = \{t^i,\ i = \overline{1,n}\}$. We represent each document t^i by the vector $w^i = (w^i_1, ..., w^i_j, ..., w^i_m)$ where w^i_j is the number of occurences of word w_j in the document t^i (see Table 6.2 below).

Teams \ Words	w_1	$\cdots$	w_j	$\cdots$	w_m
t_1	4	$\cdots$	12	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_i	2	$\cdots$	3	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	0	$\cdots$	0	$\cdots$	2

Table 6.2: Documents (teams) representation using words frequency

The distances between clusters are computed using the words frequency in the given vocabulary (i.e. the above defined vectors). For this we used a classical distance (formula 2.1) or the Jaccard dissimilarity (formula 5.1).

We used as classification algorithms the agglomerative hierarchical methods from Section 3.4.3: classical AHC, the initial 2-3 AHC [Ber02d] and our 2-3 AHC algorithms (see Chapter 3), all with the complete-link.

6.2.4 Analyses and Results

In this Section we present results from the five experiments that we performed in two analyses. In the first one, we used the classical Euclidean distance on the words frequencies for all experiments and the Jaccard coefficient for the T-C experiment. For the second analysis, we compared our 2-3 AHC algorithms with the classical AHC for the five experiments.

Since the created structures are to big to be included here, we will present only small parts of the created structures.

6.2.4.1 Experiments with the classical distance

We first compared classifications obtained in the five experiments. The motivation here was to study the classifications obtained using different parts of the activity reports.

We obtained three main results when analyzing the five experiments.

Result 1 (preprocessing):

Using the same methods, the classifications were very different from one experiment to another.

For example, our team, AxIS, was grouped in each classification in different clusters with different teams. Figures 6.1 and 6.2 present these classifications for the K-all and the T-PF experiments. As we can see, the classification using the presentation and foundations sections (T-PF experiment) is more pertinent from the research themes organization point of view. We need to mention that in the case of our team, this could be also interpreted as a consequence of the pluri-disciplinary nature of our group.

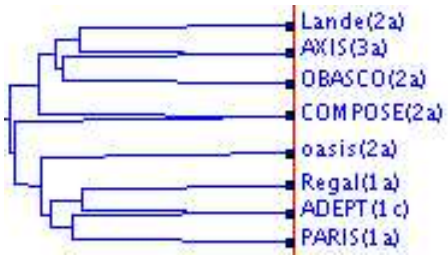


Figure 6.1: AxIS team in K-all analysis

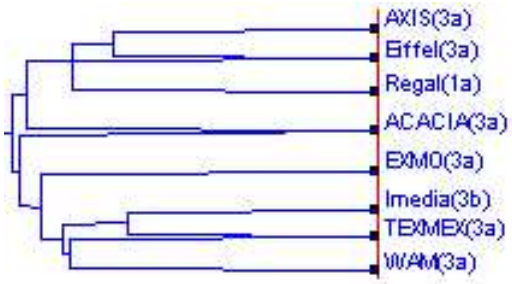


Figure 6.2: AxIS team in T-PF analysis

Result 2 (distance):

During the experiments we discovered some "atypical" teams.

For example, during the T-P analisys we found that the *epidaure* team (see Figure 6.3) used some specific words in its presentation section, words that were not used by

other teams. Moreover, from its remaining words, the word *imaging* was heavely used: 10 times in four phrases. Since the classical distance that we used takes into account the word frequency, *epidaure* became an atypical team.

Figure 6.3: Atypical teams in T-P experiment

For all keywords experiment (K-all) we also found some atypical research teams (see Figure 6.4). This was mainly due to the very small number of keywords use by the teams to describe their sections. Indeed, *Metalau* team had only one keyword in its entire activity report, while *tanc* had only two.

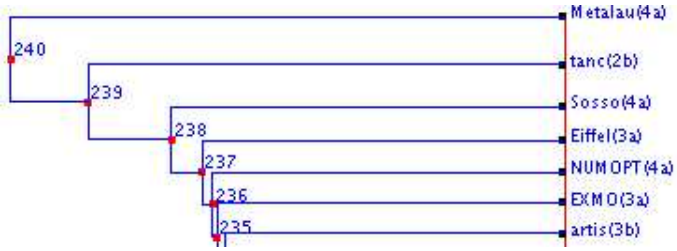


Figure 6.4: Atypical teams in K-all experiment

In the analysis of the bibliographic section, we found only one atypical team, *estime*, which had a small number of very specific keywords: *nuclear* and *supercomputing*.

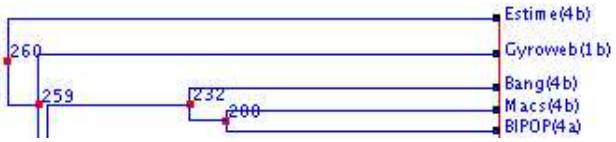


Figure 6.5: Atypical teams in T-C experiment

Result 3 (distance):

The influence of general words frequency can be avoided using Jaccard distance.

When using the classical distance on the documents frequency table (Table 6.2), we discovered that some "general" words influenced the computed distance due to their frequent use by most of the teams. These general words include for example *applications, computer, research, team,* and should be eliminated in the data preprocessing step.

Therefore, for our second analysis, we used the Jaccard coefficient [Sne57] on the document/word matrix from Table 6.2 to compute the distance between teams. Since the index uses only the presence of the words, not their frequency, we obtained different classifications (without atypical teams) compared to the first analysis and more pertinent from the research themes organization point of view (see Appendix E).

Table 6.3 presents an obtained clustering for the T-C experiment using the Jaccard coefficient, the 2-3 AHC avoiding blind merging and the complete link.

IDOPT **4b**, tropics **1a**, Ares **1b**, PLANETE **1b**, ALADIN **4b**, POPS **1a**, ScAlApplix **4b**, mistral **1b**, aces **1b**, reso **1b**, Hipercom **1b**, armor **1b**, mascotte **1b**, calvi **4b**, MADYNES **1b**, Opale **4b**	apache **1a**, Cristal **2a**, Triskell **1c**, Contraintes **2a**, A3 **1a**, calligramme **2a**, Arenaire **2b**, SECSI **2a**, LEMME **2a**, coprin **2b**, VASY **1c**, R2D2 **1a**, Moscova **1c**, Arles **1a**, AlGorille **1b**, oasis **2a**, MOSTRARE **3a**, Espresso **1c**, Runtime **1a**, mimosa **1c**, Sardes **1a**, VerTeCs **1c**, ReMaP **1a**, Regal **1a**, OBASCO **2a**, COMPOSE **2a**, JACQUARD **1a**, LogiCal **2a**, Grand-Large **1a**, caps **1a**, PARIS **1a**, Compsys **1a**, LeD **3a**, cassis **2a**, PROTHEO **2a**, ADEPT **1c**, DaRT **1c**, Lande **2a**, MIRO **2a**, modbio **2a**	REVES **3b**, i3D **3a**, in-situ **3a**, siames **3b**, Fractales **4a**, Sosso **4a**, artis **3b**, EVASION **3b**, Epidaure **3b**, Air2 **3b**, movi **3b**, IPARLA **3b**, ALCOVE **3b**, LEAR **3b**, ISA **3b**, PRIMA **3a**	cordial **3a**, Atoll **3a**, Bang **4b**, Macs **4b**, BIPOP **4a**, SIGNES **3a**	EXMO **3a**, cordial **3a**, orpailleur **3a**, Orion **3a**, Atoll **3a**, DREAM **3a**, WAM **3a**, AXIS **3a**, symbiose **3a**, MAIA **3a**, cortex **3a**, SIGNES **3a**, ACACIA **3a**
Smis **3a**, MErLIn **3a**, gemo **3a**, TEXMEX **3a**, ECOO **3a**, parole **3a**, Gyroweb **1b**	IDOPT **4b**, tropics **1a**, ALADIN **4b**, ScAlApplix **4b**, Estime **4b**, calvi **4b**, Opale **4b**	Algo **2b**, adage **2b**, geometrica **2b**, Spaces **2b**, galaad **2b**, tanc **2b**	Imara **4a**, Micmac **4b**, TRIO **1c**, e-Motion **3b**, COMORE **4a**, HELIX **3a**, sagep **4b**, macsi **4a**, icare **4a**	Ariana **3b**, VISTA **3b**, Odyssee **3b**, sigma2 **4a**, Mirages **3b**, TEMICS **3b**, ATLAS **3a**, Imedia **3b**, Metalau **4a**, IS2 **4a**, METISS **3a**
CONGE **4a**, Mathfi **4b**, Mirages **3b**, Miaou **4a**, Cafe **2b**, Trec **1b**, Metalau **4a**, IS2 **4a**	s4 **1c**, tick **1c**, Trisell **1c**, SECSI **2a**, VASY **1c**, Espresso **1c**, Ostre **1c**, VerTeCs **1c**, cassis **2a**, Lande **2a**	MErLIn **3a**, Eiffel **3a**		

Table 6.3: 2-3 AHC V3 clustering in T-C experiment using Jaccard and CL

6.2.4.2 Comparaison of the classical AHC and 2-3 AHC

Result 4 (2-3 AHC):

The 2-3 AHC avoiding the blind merging (2-3 AHC V3) was the only 2-3 AHC algorithm with a positive gain compared to the classical hierarchy.

For the five experiments, we compared the created 2-3 hierarchies with the classical hierarchies (using complete link). The objective here is similar to the one in Section 3.5: to perform a qualitative comparaison between the classical AHC and 2-3 AHC.

The Stress gain (see Section 3.5) was used to compare the resulting structures. As we can see in Table 6.4, the 2-3 AHC avoiding the blind merging (2-3 AHC V3) was the only 2-3 AHC algorithm with a positive gain when compared to the classical hierarchy.

	K-F	**K-all**	**T-P**	**T-PF**	**T-C**
2-3AHC ref V2	-55.14%	-38.54%	-69.02%	-22.08%	-64.04%
2-3AHC V2	-56.03%	-37.96%	-98.07%	-19.69%	-65.71%
2-3AHC (ref) V3	*6.52%*	*11.15%*	*3.56%*	*6.64%*	*11.45%*
2-3AHC ini	-49.51%	-37.1%	-104.19%	-29.1%	-65.71%

Table 6.4: Complete-link Stress gains of 2-3 AHC using the simple distance

A small example of information gain between the 2-3 AHC V3 and the classical AHC in the T-P experiment, is given in Figures 6.6 and 6.7. Here we can see that for example the *mascotte* team is the intermediary element between the other two classes: $\{mistral, reso, Ares, PLANETE\}$ and $\{macsi, Trec, icare\}$.

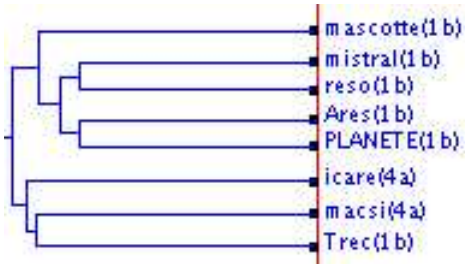


Figure 6.6: AHC in T-P experiment

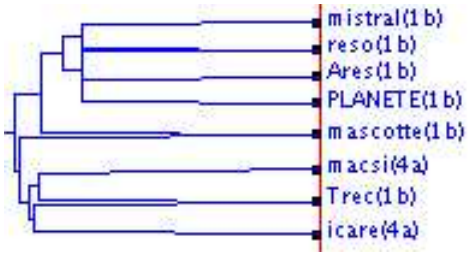


Figure 6.7: 2-3 AHC in T-P experiment

6.3 Other Applications

In this section we briefly describe two other applications of the 2-3 AHC in Documents Clustering. Our Hierarchical Clustering Toolbox presented in Section 4.2 was used in these applications.

INEX Movie DB

In [Gar05], the authors analyzed a collection of XML documents from the Document Mining Track* of the INEX† Initiative. They used only the structure information of the XML documents, not their content, in order to find different "structural families of documents". The purpose was to characterize each predefined cluster in terms of frequent "structural" patterns and then to classify ordered labeled trees.

To extract the needed sequential patterns, they used the PSP algorithm [MCP98]. Then the sequential patterns, were grouped using an 2-3 AHC algorithm from the HCT toolbox. The used dissimilarity was defined by the following formula:

$$D(S_1, S_2)) = 1 - \frac{2|LCS(S_1, S_2)|}{|S_1| + |S_2|}$$

where S_1, S_2 are the mined sequences, $LCS(S_1, S_2)$ is the Longest Common Subsequence and $|S|$ is the length of the sequence S.

The authors then validated the obtained clustering against the predefined categories given in the Document Mining Track

Sanskrit Documents

In [Tan05b], our motivation was to compare old Sanskrit documents having all a common origin, in order to create a critical edition and/or a philogenetic tree. This work was done in the context of an EuropAID project.

They used a LCS algorithm to detect inversions in the analyzed documents and a distance measure based on a spreading function. Our HCT toolbox was then used to cluster a small set of documents.

6.4 Discussion and Perspectives

In this chapter our 2-3 AHC algorithms were used in the field of the XML Document Clustering. We clustered INRIA's research teams by analyzing their 2003 activity reports (semi-structured XML documents).

*`http://xmlmining.lip6.fr/Home`

†INEX: INitiative for the Evaluation of XML Retrieval
`http://inex.is.informatik.uni-duisburg.de/2005`

We first selected different parts (sub-structures) of XML documents for the clustering tasks. Then, a second level of selection was performed for a linguistic selection of words within the text.

We found that different parts of the activity reports produce different classifications. Some classifications (i.e. keywords, presentation) are especially influenced by the flexibility of the content: the use of keywords, repetition of words, etc. In this context, a better data preprocessing should be performed to eliminate some "general" words.

We used the Jaccard coefficient to compare our classification with the existing research teams organization. This analysis was performed mainly to avoid the preprocessing inconvenients and to compare the resulting classification with the existing research organization. In this case better results were obtained compared with the first analyses using the classical distance on the words frequencies.

In a second analysis, for the five performed experiments we compared the obtained classical hierarchies with the 2-3 hierarchies obtained by our 2-3 AHC algorithms. As expected, the 2-3 AHC avoiding the blind merging had the best results when using the Stress gain presented in Section 3.5.

Our future work concerns:

- a detailed comparaison with other similar works, such as the one in [DLTV05];
- a better vocabulary (words) extraction, based on an ontology defined on INRIA's research topics;
- the use of other distance measures on the obtained keywords.

Conclusions and Perspectives

Chapter 7

Conclusions and Perspectives

We summarize in this Chapter our main contributions on the Agglomerative 2-3 Hierarchical Classification study and implementation that we realized during this thesis. Some future works and perspectives are presented in the end of this Chapter.

7.1 Main Contributions of this Thesis

During this work, we have proposed, analyzed and validated a new Agglomerative 2-3 Hierarchical Classification algorithm based on the one introduced in [Ber02d].

The three main contributions of this thesis can be summarized as follows:

1. We performed a theoretical study of the 2-3 hierarchies and of the initial 2-3 AHC algorithm. This allowed us to discover new properties of the 2-3 hierarchies and a special case of merging;

2. We proposed a new 2-3 AHC algorithm with a lower complexity than the initial one from [Ber02d]. We validated our 2-3 AHC algorithm's complexity and we compared its results with the classical AHC's ones. We also developed the `Hierarchical Clustering Toolbox` to ease the results interpretation and comparison;

3. We studied the applicability of the 2-3 AHC in two Data Mining fields: Web Usage Mining and Document Clustering. The 2-3 AHC was also integrated in the CBR*Toolbox* as an indexing method, to be used in the future in different recommender systems like the Be-TRIP recomender system that we proposed for the mobility contexts.

*Case Based Reasoning Toolbox: http://www-sop.inria.fr/axis/cbrtools

7.1.1 Theoretical Study of the 2-3 AHC

In the first part of our study described in Chapter 2, we focused on the theoretical properties of the 2-3 hierarchies and of the initial 2-3 AHC algorithm introduced in [Ber02d].

We have revealed four new properties of the 2-3 hierarchies that were later use to formulate our new 2-3 AHC algorithm. These properties allowed us to propose a new merging step for 2-3 AHC algorithm, denoted the *intermediate merging* step. Also, using these properties we specified exactly the clusters to be eliminated from the candidate set in the `Update` step and the dissimilarities to compute. This contributed in the 2-3 AHC algorithm complexity reduction.

We introduced an "on-the-fly" refinement to replace the recursive refinement step from the end of the 2-3 AHC algorithm. We called it the *integrated refinement*, and since it influences the created structure we made this step optional to diversify the outputs that can be obtained using our 2-3 AHC algorithm.

Since the proper intersection is the main characteristic of the 2-3 hierarchies, we studied its influence on the aggregation index, algorithm execution, etc. We thus defined the "best" choices for indexing formulas and aggregation indexed for the complete and single link.

We revealed a particular case of merging that we called the *blind merging*. We proposed a solution to avoid these kind of mergings, which creates different structures too. Therefore, avoiding the blind mergins was also integrated as an option in our 2-3 AHC algorithm.

7.1.2 A New 2-3 AHC Algorithm

Based on the theoretical study of the 2-3 hierarchies and of the initial 2-3 AHC algorithm, we proposed a new 2-3 AHC algorithm wih a reduced complexity of $\mathcal{O}(n^2 \log n)$ compared to $\mathcal{O}(n^3)$ in the initial one. The tests on different datasets, real and generated, confirmed our theoretical complexity analysis.

The principle of this new 2-3 AHC algorithm is similar to the one of the classical AHC algorithm.

Having four 2-3 AHC algorithm variants given by the two optional steps of the algorithm, we performed a "qualitative" analysis by comparing the results to the classical AHC algorithm's ones on real and large datasets.

First, we compared the number of created clusters for our 2-3 AHC algorithm and the classical one. The results were satisfactory as we found that we can obtain with our

algorithm up to 50% more clusters using the complete link and 45% using the single link. We saw on smaller real datasets (Ruspini, urban itineraries) that this larger number of created clusters, can provide us with a richer structure, i.e. more information for results interpretation.

Next, we analyzed the "quality" of the created structures (hierarchies and 2-3 hierarchies) by comparing their induced dissimilarity matrices with the analyzed dissimilarity matrix. The matrices were compared using the Stress [JW82] index that measures the degree of correspondence between them. This gave us an estimation of how "different" the induced matrices were compared to the initial data.

We found that the Stress gain was very variable, and for some of the created 2-3 hierarchies there was even a loss of information compared to the classical hierarchy. However, our 2-3 AHC algorithm avoiding the blind merging was the only one to always produce Stress gains compared to the classical hierarchies (e.g "better structures"). The maximum Stress gain reached in this case 84% for the 2-3 AHC algorithms*.

This lead us to conclude that the 2-3 AHC algorithm avoiding the blind merging is the most "stable" one, although its Stress gains were also variable, and should be the one to chose if one wants to assure the construction of a richer 2-3 hierarchy (than the classical hierarchy) on a give dataset. However, when possible, one could chose to execute all possible 2-3 AHC algorithms (including the initial proposed in [Ber02d]), compare their gains, and chose the one that represents the best the analyzed data.

We compared our 2-3 AHC with the Ascendent Pyramidal Classification (APC) algorithm [EUR]. For this we used the SODAS software [Did02], but due to the different implementations and a wrong results computation in SODAS, a direct comparison was not possible.

We performed a first implementation (in Java) of an 2-3 AHC algorithm, based on the object-oriented model that we also propose. This object-oriented model was then used to design and implement `Hierarchical Clustering Toolbox` and to integrate the 2-3 AHC into the CBR*Tools framework.

Hierarchical Clustering Toolbox

The object-oriented model of our algorithm was used in a visualization and analysis toolbox called the `Hierarchical Clustering Toolbox`. The classical AHC and the initial 2-3 AHC algorithms were integrated as well.

The toolbox was designed to ease the results interpretation and to compare the different hierachical methods using different "quality" criteria or a direct visual analysis of the structures.

*The 84% gain is for more than 10 analyzed objects, ($n \geq 10$) since for smaller n (i.e. $n = 3$) the gain can be 100%

The toolbox was also used by other members of the team in different analyses [GMT05, Tan05b, TMT06].

CBR*Tools integration

We integrated our object-oriented model of the 2-3 AHC algorithm in the Case-Based Reasoning object-oriented framework, CBR*Tools [Jac98], as an indexing method. The purpose of this integration was to use our method for case indexing in a recommender system like the Be-TRIP mobility recomender system that we proposed in [CGT04, TCG04]. Due to its high complexity, only off-line analysis can be performed so far.

7.1.3 2-3 AHC Applications

After testing the 2-3 AHC on different datasets, we used our 2-3 AHC algorithm in the context of two Data Mining applications:

- a Web Usage Mining application on INRIA's research teams using the visited topics on its Web sites;
- a Document Mining application on INRIA's research teams using the XML collection of INRIA's 2003 activity reports.

Web Usage Mining application

We analyzed the impact of INRIA's Web site structure on users navigations. This was done on two time periods: before and right after the reorganization of the research teams into research themes.

The data from two of INRIA's Web sites was used. The Web logs data was preprocessed using the methodology proposed in [Tan05a] which we improved for URL generalization and keywords extraction. We classify the research teams based on the topics visited by the Web users in their search for information.

The 2-3 AHC produced interesting results, richer than the classical AHC and better than the ones obtained in [GCGR$^+$04]. Although the second analyzed period was shortly after the change, we have found that usually users navigations are influenced by the Web site structure, which corresponds to the research teams organization.

Document mining application

We clustered INRIA's research teams based on their 2003 activity reports. This XML collection of activity reports was preprocessed using the methodology proposed in

[DLTV05]. Since the XML documents were semi-structured, we used beside a linguistic selection of words, a selection at the structure level. Thus we found that different parts of the activity reports produce different classifications and that some improvements can be made in the preprocessing step.

As in the previous analyses form Section 3.5, the 2-3 AHC avoiding the blind merging obtained the best results, and moreover the only positive Stress gains compared to the classical AHC.

7.2 Future Works and Perspectives

There are several perspectives opened by this research and they concern on one hand the 2-3 AHC algorithm and results analysis and on the other hand the 2-3 AHC as a part of the HCT and CBR*Tools.

The future works and perspectives concerning the 2-3 AHC algorithm and results analysis can be summarized as follows:

- deeper study of other aggregation indexes such as the average linkage or the Ward criterion for the particular case of properly intersecting clusters;
- a deeper structure refinement for large structures. This is especially useful for visual results interpretation which is very difficult when the created clusters number increases (more than 100 for example). One can use for example a threshold in the difference of clusters indexing level (f value) combined with a measure of the clusters "homogeneity" as in [RD05];
- definition of other "quality" measures to compare the created structures (classical hierarchies and 2-3 hierarchies) on same datasets. One could use in this case a measure based on the clustering accuracy, when possible.
- an automatic partitioning level to cluster the initial elements using the level of the created clusters.
- more tests are necessay between 2-3 AHC and APC using similar and updated implemetations: on induced dissimilarities, using same programming language (implementations), etc.

For the Hierarchical Clustering Toolbox (see Section 4.2), the future works and perspectives concern:

- the integration of a general DB Explorer feature, used to explore the available data in DB and in which custom queries for selecting data can be performed, as in [Tan05a].
- the extension of the supported data formats (e.g. arff files used in the WEKA software [WF05]).

Future work regarding the CBR*Tools [Jac98] integration of our 2-3 AHC algorithm include the study of an automatic partitioning criteria and eventually an incremental feature for the hierarchical algorithms to allow on-line analysis.

Bibliography

[ADLCR99] J.C. Aude, Y. Diaz-Lazcoz, J.J. Codani, and J.L. Risler. Applications of the pyramidal clustering method to biological objects. *Computer & Chemistry*, 23(3-4):303–15, 15 June 1999.

[AS95] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. 11th Int. Conf. Data Engineering, ICDE*, pages 3–14. IEEE Press, 6–10 1995.

[Aud99] J.-C. Aude. *Analyse de génomes microbiens : apport de la classification pyramidale*. PhD thesis, Univ. Paris 9 - Dauphine, France, 1999.

[Ban92] H-J. Bandelt. Four-point characterization of the dissimilarity function obtained from indexed closed weak hierarchies. Mathematisches Seminar, Hamburg Universitat, 1992.

[Bar01] G. Bartolini. Web usage mining and discovery of association rules from HTTP servers logs. Monash University, Melbourne, 2001.

[Bat88] A. Batbedat. Les isomorphismes hts et hte (après la bijection de benzécri-johnson). *Metron*, 46:47–59, 1988.

[BBO04] J.-P. Barthélemy, F. Brucker, and C. Osswald. Combinatorial optimization and hierarchical classifications. *A Quarterly Journal of Operations Research*, 2(3):179 – 219, 2004.

[BD89] H-J. Bandelt and W.M. Dress. Weak hierarchies associated with a similarity measure - an additive clustering technique. *bulletin of Mathematical Biology*, 51(1):133–166, 1989.

[BD94] H-J. Bandelt and W.M. Dress. An order theoretic framework for overlapping clustering. *Discrete Math*, 136:21–37, 1994.

[BD00] H.H. Bock and E. Diday. *Analysis of Symbolic Data*. Study in Classification, Data Analysis and Knowledge Organisation. Springer Verlag, 2000.

[Ben73] J.-P. Benzécri. *L'Analyse des Données*. DUNOD, Paris, 1973.

[Ben00] A. Benedek. Artificial Neural Network based memory indexing models for CBR case base. DEA Stage Report, 2000. Universite de Nice-Sophia Antipolis.

[Ber] P. Bertrand. System of sets such that each set properly intersects at most one other set - application to cluster analysis. *Discrete Applied Mathemathics*. Accepted for publication, To appear.

[Ber86] P. Bertrand. Etude de la representation pyramidale. Universite Paris IX-Dauphine, 1986. These de 3^{eme} cycle.

[Ber02a] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[Ber02b] P. Bertrand. A new presentation of the algorithm of 2-3 HAC, 11 may 2002. Internal AxIS document.

[Ber02c] P. Bertrand. Indicages, 11 may 2002. Internal AxIS document.

[Ber02d] P. Bertrand. Set systems for which each set properly intersects at most one other set - Application to Cluster Analysis. Research Report Ceremade 0202, Universite Paris-9, France, 2002.

[BHT05] E. Barbu, P. Héroux, and E. Trupin. Classification non supervisée hiérarchique incrémentale basée sur le calcul de dissimilarités. In *Comptes rendus des 12-èmes Rencontres de la Société Francophone de Classification*, 2005.

[BJ97] P. Bertrand and M.F. Janowitz. Pyramids and weak hierarchies in the ordinal model for clustering. Technical Report 9709, Ceremade (URA CNRS 749), 1997.

[BJ03] P. Bertrand and M.F. Janowitz. The k-weak hierarchical representations: an extension of the indexed closed weak hierarchies. *Discrete Applied Mathemathics*, 127(2):199–220, April 2003.

[BPZKC00] V. Batagelj, E. Pavletic, M. Zavers(nik, and S. Korenjak-Cerne. Clustering large datasets and visualizations of large hierarchies and pyramids: Symbolic data analysis approach. In *Workshop on Symbolic Data Analysis: Theory, Software and Applications for Knowledge Mining, PKDD 2000*, 2000.

[Bri91] P. Brito. *Analyse de données symboliques: Pyramides d'héritage*. PhD thesis, Universite Paris 9 Dauphine, 1991.

[Bri02] P. Brito. Hierarchical and pyramidal clustering for symbolic data. *Journal of the Japanese Society of Computational Statistics*, 2002.

[Bru01] F. Brucker. *Modèles de classification en classes empiétantes*. PhD thesis, E.H.E.S.S., 2001.

[Bru05] F. Brucker. Inférieures-maximales faiblement hiérarchiques. In *Comptes rendus des 12-èmes Rencontres de la Société Francophone de Classification*, Montréal, Canada, 30 May - 1 June 2005.

[Bus05] R. Busseuil. Classification des itinéraires pour l'aide à la navigation assistée par gps. Master's thesis, ENS Cachan, 2005.

[CBT04] S. Chelcea, P. Bertrand, and B. Trousse. Un Nouvel Algorithme de Classification Ascendante 2-3 Hiérarchique. In *Reconnaissance des Formes et d'Intelligence Artificielle (RFIA 2004)*, Centre de Congrès Pierre BAUDIS, Toulouse, 28-30 Janvier 2004.

[CDSL+05] S. Chelcea, A. Da Silva, Y. Lechevallier, D. Tanasa, and B. Trousse. Benefits of intersite pre-processing and clustering methods in e-commerce domain. In Petr Berka and Bruno Crémilleux, editors, *Proceedings of the ECML/PKDD2005 Discovery Challenge, A Collaborative Effort in Knowledge Discovery from Databases*, pages 15–21, Porto, Portugal, 3-7 October 2005.

[CGT04] S. Chelcea, G. Gallais, and B. Trousse. Recommandations personnalisées pour la recherche d'information facilitant les déplacements. In *Premières Journées Francophones : Mobilité et Ubiquité 2004*, pages 143 – 150, ESSI, Nice, Sophia-Antipolis, France, 1-3 June 2004. Cepadues - ISBN : 2-85428-653-7 / ACM Digital Library - ISBN : 1-58113-915-2.

[CMS97] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[CT04] S. Chelcea and B. Trousse. Application of the 2-3 agglomerative hierarchical classification on web usage data. In Dana Petcu, Viorel Negru, Daniela Zaharie, and Tudor Jebelean, editors, *Proceedings of SYNASC 2004, 6th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, pages 107–118, Timisoara, Romania, 26-30 September 2004. Mirton Publisher, ISBN 973-661-441-7. ISBN 973-661-441-7.

[CT05] S. Chelcea and B. Trousse. Classification 2-3 hiérarchique de données du web. In Suzanne Pinson and Nicole Vincent, editors, *Actes des 5èmes journées Extraction et Gestion des Connaissances (EGC'2005)*, volume 1 of *Revue des Nouvelles Technologies de l'Information (RNTI-E-3)*, page 219. Cépaudès-Editions, ISBN 2.85428.677.4, January 2005. poster.

[DBM01] E. Diday, P. Bertrand, and E. Mfoumoune. Classification pyramidale : Une nouvelle implémentation de la cap, 2001.

[Des04] T. Despeyroux. Practical semantic analysis of web sites and documents. In *The 13th World Wide Web Conference, WWW2004*, New York City, USA, 17-22 May 2004.

[DF94] J. Diatta and B. Fichet. *From Asprejan hierarchies and Bandelt-Dress weak-hierarchies to quasi-hierarchies.* Springer, berlin Heidelberg New York, 1994.

[Dia96] J. Diatta. *Une extension de la classification hiérarchique : les quasi-hiérarchies.* PhD thesis, Université de Provence, France, 1996.

[Dia97] J. Diatta. Dissimilarités multivoies et généralisation d'hypergraphes sans triangle. *Mathématiques, Informatique et Science Humaines*, 138:57–73, 1997.

[Did73] E. Diday. The dynamic cluster method in non-hierarchical clustering. *J. Comput. Inf. Sci.*, 2:61–88, 1973.

[Did84] E. Diday. Une représentation visuelle des classes empietantes: les pyramides. Technical Report 291, INRIA, Rocquencourt 78150, France, 1984.

[Did86] E. Diday. Orders and overlapping clusters by pyramids. In J.D. De Loeuw and Et al, editors, *Proceed. Multidimensional Data Analysis*, Leiden, The Netherlands, 1986. DSWO Press.

[Did87] E. Diday. Introduction l'approche symbolique en analyse des données. In *Premiere Journees Symbolique-Numerique.* Universite Paris IX Dauphine, December 1987.

[Did02] E. Diday. An introduction to symbolic data analysis and the SODAS software. *Journal of Symbolic Data Analysis, International Electronic Journal*, 1(1), 2002.

[DLTV05] T. Despeyroux, Y. Lechevallier, B. Trousse, and A.-M. Vercoustre. Experiments in clustering homogeneous xml documents to validate an existing

typology. In *Proceedings of the 5th International Conference on Knowledge Management (I-Know)*, number 1, Vienne, Autriche, July 2005. Journal of Universal Computer Science.

[Dub87] R. Dubes. How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645–663, 1 November 1987.

[EUR] Project EUROSTAT. Sodas software. http://www.ceremade.dauphine.fr/ touati/sodas-pagegarde.htm.

[Fas99] D. Fasulo. An Analysis of Recent Work on Clustering Algorithms. Technical Report 01-03-02, Department of Computer Science & Engineering, University of Washington, Seattle, WA, April 26 1999.

[Fic84] B. Fichet. Sur une extension de la notion de hiérarchie et son équivalence avec quelques matrices de robinson. In *Actes des Jourées de statistique de la Grande Motte*, pages 12–12, 1984.

[FPSSU96] U.M. Fayyad, G. Piatetsky-Shapiro, Padhraic Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[FSS00] Yongjian Fu, K. Sandhu, and M. Shih. A generalization-based approach to clustering of web usage sessions. In *Proc. 1999 KDD Workshop Web Mining*, volume 1836 of *LNCS*, pages 21–38. Springer-Verlag, 2000.

[Gar05] Calin Garboni. Sequential pattern mining for structure-based xml document classification. Master's thesis, West University of Timisoara, Romania, 2005.

[GCA98] A Gil, C. Capdevila, and A. Arcas. On the efficiency and sensitivity of a pyramidal classification algorithm. Economics working paper 270, Barcelona, 1998.

[GCGR+04] A. El Golli, B. Conan-Guez, F. Rossi, D. Tanasa, B. Trousse, and Y. Lechevallier. Les cartes topologiques auto-organisatrices pour l'analyse des fichiers logs. In *11èmes Rencontre de la Société Francophone de Classification*, Bordeaux, 8-10 septembre, 2004. to appear.

[GH85] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.

[GHJ91] A. Guénoche, P. Hansen, and B. Jaumard. Efficient algorithms for divisive hierarchical clustering with the diameter criterion. *Journal of Classification*, 8:5–30, 1991.

[GMT05] C. Garboni, F. Masseglia, and B. Trousse. Sequential pattern mining for structure-based xml document classification. In *Collection of Fourth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2005)*, Schloss Dagstuhl, Germany, November 2005.

[Gor99] A.D. Gordon. Classification. Chapman and Hall, 2nd ed., 1999.

[GS94] Wolfgang Gaul and M. Schader. Pyramidal classification based on incomplete dissimilarity data. *Journal of Classification*, 11:171–193, 1994.

[Har83] G. Hart. The occurrence of multiple upgma dendrograms. *Numerical Taxonomy*, pages 254–8, 1983.

[Jac98] M. Jaczynski. *Modelé et plate-forme à objets pour l'indexation des cas par situations comportementales : application à l'assistance a la navigation sur le Web*. PhD thesis, Universite de Nice - Sophia Antipolis, December 1998.

[JD98] A.K. Jain and R. Dubes. *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, 1998.

[JF88] R. E. Johnson and B. Foote. Designing Reusable Classes. *Journal of Object-Oriented Programming*, 1:22–35, June 1988.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JS68] N Jardine and R Sibson. The construction of hierarchic and non-hierarchic classifications. *Computer Journal*, 11:177–184, 1968.

[Jul02a] L. Jullien. 2-3 hierarchies, 2-3 ultrametriques, algorithme 2-3 cah. DEA Stage Report, 19 septembre 2002. Universite Paris I.

[Jul02b] L. Jullien. Merging step dans l'algorithm 2-3 HAC, May 2002. Internal AxIS document.

[JW82] A.R. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*, chapter 12. Prentince Hall, 1982.

[Koh01] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2001.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, Inc., March 1990.

[KTP97] N.I. Karacapilidis, B. Trousse, and D. Papadias. Using case-based reasoning for argumentation with multiple viewpoints. In D. Leake and E. Plaza, editors, *Case-Based Reasoning Research and Development, Proceedings of the 2nd Int. Conference on Case-Based Reasoning (ICCBR-97)*, volume 1266 of *Lecture Notes in AI*, pages 541–552, Providence, Rhode Island, July 1997. Springer-Verlag, Berlin.

[Ler97] I.-C. Lerman. Comparing classification tree structures : a special case of comparing q-ary relations. Technical Report RR-3167, INRIA, 1997.

[LL95] F.-J. Lapointe and P. Legendre. Comparison tests for dendrograms : A comparative evaluation. *Journal of Classification*, 12:265–282, 1995.

[LMTT06] Y. Lechevallier, F. Masseglia, D. Tanasa, and B. Trousse. Gwum : une généralisation des pages web guidée par les usages. In *INFORSID 2006*, Hammamet, Tunisie, 1-3 June 2006. to appear.

[LN99] B. Lavoie and H. F. Nielsen. Web characterization terminology & definitions sheet. http://www.w3c.org/1999/05/WCA-terms/, May 1999.

[LW67] G.N. Lance and W.T. Williams. A general theory of classification sorting strategies. *Computer Journal*, 9:373–380, 1967.

[LW03] C. Li and W. H. Wong. *DNA-Chip Analyzer (dChip)*. Spinger, 2003.

[Mac67] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[Mat] MathWorks. Bioinformatics toolbox. http://www.mathworks.com/products/bioinfo/.

[MC85] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[MCP98] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol. 1510*, pages 176–184, Nantes, France, September 1998.

[Mfo98] E. Mfoumoune. *Les aspects algorithmiques de la classification ascendante pyramidale et incrementale*. PhD thesis, Universite Paris 9 Dauphine, 1998.

[MJHS96] B. Mobasher, N. Jain, E.H. Han, and J. Srivastava. Web Mining: Pattern Discovery from World Wide Web Transactions. Technical Report TR-96050, University of Minnesota, Department of Computer Science, Minneapolis, 1996.

[MPC99] F. Masseglia, P. Poncelet, and R. Cicchetti. An Efficient Algorithm for Web Usage Mining. *Networking and Information Systems Journal (NIS)*, 2(5-6):571–603, 1999.

[NK02] O. Nasraoui and R. Krishnapuram. An evolutionary approach to mining robust multi-resolution web profiles and context sensitive url associations. *Int. Journal of Computational Intelligence and Applications*, 2(3):339–348, 2002.

[Phi71] J.B. Phipps. Dendrogram topology. *Systematic Zoology*, 20:306–308, 1971.

[PHMAZ00] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining Access Patterns efficiently from Web logs. In *Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00)*, Kyoto, Japan, April 2000.

[Pod02] J. Podani. Simulation of random dendrograms and comparison tests: some comments. *Journal of Classification*, 17:123–142, 2002.

[Rao71] M.R. Rao. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 66(335):622–626, 1971.

[RD04] M. Rahal and E. Diday. La classsification pyramidale symbolique : Selection de paliers et de variables. In *Actes des 11emes Rencontres de la SFC*, pages 146–149, Bordeaux, 8-10 September 2004.

[RD05] M. Rahal and E. Diday. Elagage et aide à l'interprétation symbolique et graphique d'une pyramide classifiante. In N. Vincent and S. Pinson, editors, *Actes des 5ème journées Extraction et Gestion des Connaissances (EGC 2005), Revue des Nouvelles Technologies de l'Information (RNTI-E-3)*, volume I, pages 135–146, Paris, France, January 2005. Cépaduès-Editions.

[Rus69] E. H. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, 1969.

[SA79] R.N. Shepard and P. Arabie. Additive clustering: representation of similarities as combination of discrete overlapping properties. *Psyhological Review*, 86:87–123, 1979.

[SA96] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[Sam95] W. Sam. *Extending and benchmarking Cascade-Correlation.* PhD thesis, Computer Science Department, University of Tasmania, 1995.

[SBK01] C. Shahabi and F. Banaei-Kashani. A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking. In Book Chapter, 2001. WebKDD'01.

[Sch94] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

[Sch97] H.A. Schmid. Systematic framework design by generalization. *Communications of the ACM*, 40(10):48–51, 1997.

[SCH+01] M. Shyu, S. Chen, C. Haruechaiyasak, C. Shu, and S. Li. Disjoint Web Document Clustering and Management in Electronic Commerce. In *Seventh International Conference on Distributed Multimedia Systems (DMS'2001)*, pages 494–497, Tamkang University, Taipei, Taiwan, September 26-28 2001.

[Sib73] R Sibson. Slink: an optimally efficient algorithm for the single link cluster method. *Computer Journal*, 16:30–34, 1973.

[Sne57] P. H. A. Sneath. Some thoughts on bacterial classification. *Journal of General Microbiology*, 17:184–200, 1957.

[SR62] R. R. Sokal and F. J. Rohlf. The comparison of dendrograms by objective methods. *TAXON*, XI(2):33–40, 1962.

[SS63] R.R. Sokal and P.H.A. Sneath. Principles of numerical taxonomy. Freeman, San Francisco, 1963.

[SS73a] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy.* W.H. Freeman, San Francisco, 1973.

[SS73b] P.H.A. Sneath and R.R. Sokal. Numerical taxonomy. Freeman, San Francisco, 1973.

[SS02] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35(7):80–86, July 2002.

[ST05] F. Sousa and J. Tendeiro. A validation methodology in hierarchical clustering. In *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA 2005)*, 17 May 2005.

[Tan05a] D. Tanasa. *Web Usage Mining: Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support*. PhD thesis, University of Nice Sophia Antipolis, 3 June 2005.

[Tan05b] S. Tandabany. Elaborating a distance for clustering homogeneous sanskrit documents. Master's thesis, ENS Lyon, 2005.

[TCG04] B. Trousse, S. Chelcea, and G. Gallais. Faciliter les déplacements par des recommandations personnalisées à la recherche d'information. *Revue Génie Logiciel, rubrique Systèmes d'informations et transports*, 70:48 – 57, September 2004.

[Tea95] NCSA HTTPd Development Team. Ncsa httpd logoptions directive. http://hoohoo.ncsa.uiuc.edu/docs/setup/httpd/LogOptions.html, 1995.

[TJK99] B. Trousse, M. Jaczynski, and R. Kanawati. Une approche fondée sur le raisonnement à partir de cas pour l'aide à la navigation dans un hypermédia. In J-P. Balpe, S. Natkin, A. Lelu, and I. Saleh, editors, *Proceedings of Hypertexte & Hypermedia : Products, Tools and Methods (H2PTM'99)*, pages 13–26. hermes, august 1999. Paris.

[TMT06] D. Tanasa, F. Masseglia, and B. Trousse. Gwum: Usage-driven web page generalization. In *17th International Conference on Database and Expert Systems Applications, DEXA 2006*, Krakow, Poland, 4 September 2006. Soumission.

[TT01] D. Tanasa and B. Trousse. Web Access pattern Discovery and Analysis based on Page Classification and on Indexing Sessions with a Generalised Index Tree. In *SYNASC 2001, Timisora, Roumanie*, pages 62–72, october 2001.

[TT03] D. Tanasa and B. Trousse. Le prétraitement des fichiers logs web dans le "web usage mining" multi-sites. In *Journées Francophones de la Toile (JFT'2003)*, Tours, July 2003.

[TT04] D. Tanasa and B. Trousse. Advanced data preprocessing for intersites web usage mining. *IEEE Intelligent Systems*, 19(2):59–65, March-April 2004.

[TTM04] D. Tanasa, B. Trousse, and F. Masseglia. Classer pour découvrir: une nouvelle méthode d'analyse du comportement de tous les utilisateurs d'un

site web. In Georges Hébrail, Ludovic Lebart, and Jean-Marc Petit, editors, *Revue des Nouvelles Technologies de l'Information (RNTI), numéro spécial Extraction et Gestion des Connaissances (EGC'2004)*, volume 2, pages 549–560. Cépaudès-Editions, January 2004.

[W3C95] W3C. Logging control in w3c httpd. http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format, July 1995.

[War63] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of American Statistical Association*, 58(301):236–244, 1963.

[WC71] W.T. Williams and H.T Clifford. On the comparaison of two classifications of the same set of elements. *Taxon*, 20:519–522, 1971.

[WF05] I. H. Witten and E. Frank, editors. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, June 2005.

[Wil88] P. Willett. Recent trends in hierarchical document clustering: A critical review. *Information Processing & Management*, 24:577–597, 1988.

[YKM99] Fu Y., Sandhu K., and Shih M. Fast Clustering of Web Users Based on Browsing Patterns. In *World Multiconference on Systemics, Cybernetics and Informatics*, volume 5, pages 560–567, Orlando, FL, August 1999.

[You04] G. Youness. *Contributions à une méthodologie de comparaison de partitions*. PhD thesis, L'université Paris 6, 2004.

[YP97] Yiming Yang and Jan O. Pederson. A comparative study on feature selection in text categorisation. In *Proceedings of Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[ZE98] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM SIGMOD Conference*, pages 103–114, Montreal, Quebec, Canada, June 4-6 1996. University of Wisconsin-Madison.

[ZXH98] O.R. Zaiane, M. Xin, and J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Advances in Digital Libraries*, pages 19–29, Santa Barbara, CA, 1998.

Appendix A

Example of the *Blind Merging*'s Influence

We present here a small example of information loss on the resulting 2-3 hierarchy compared with the classical hierarchy, when we perform a blind merging. The data set is a set of five points with the distance matrix d_{ini} given in Figure A.1.

	b	c	d	e
a	[1]	1	1.1	1.4
b		2	1.3	1.3
c			1.3	3
d				1.2

Figure A.1: The data set

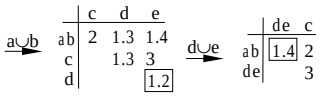


Figure A.2: AHC dissimilarities

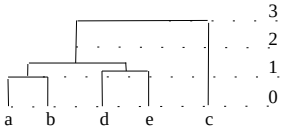


Figure A.3: The classical hierarchy

We will use here the complete-link with all its possible definitions (1), (2) and (3) from Section 2.4.2.1, the extended indexing formula:

$$f(X \cup Y) = max\{f(X), f(Y), \mu(X, Y)\} \tag{A.1}$$

from Section 2.5 and the algorithm executions, normal (**i**) and with integrated refinement (**ii**) from Section 2.7. Although we have seen that the first dissimilarity definition (1) is recommended (see Section 2.4.2.1 and 2.8), we will analyze all three cases.

For the classical hierarchical case presented in Figure A.3, we first merge $\{a\}$ and

$\{b\}$ as they are at minimum dissimilarity, $\mu = 1$, cf. Figure A.2. Next, $\{d\}$ and $\{e\}$ are merged at the dissimilarity $\mu = 1.2$. The resulting cluster $\{de\}$ will be merged with $\{ab\}$ at dissimilarity $\mu = 1.4$ and the remaining singleton, $\{c\}$, will be finally merged with the last formed cluster $\{abde\}$ at $\mu = 3$. The resulting hierarchy is presented in Figure A.3 while the resulting ultrametric in Figure A.6.

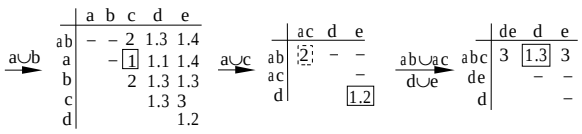


Figure A.4: 2-3 AHC dissimilarities

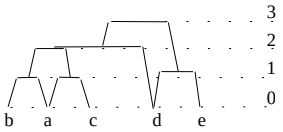


Figure A.5: The created 2-3 hierarchy

For the 2-3 AHC algorithm, we first merge $\{a\}$ and $\{b\}$, and then $\{a\}$ and $\{c\}$ having thus two clusters that properly intersects themselves (Figure A.4 and Figure A.5). At this moment we can use the intermediate refinement (**i**) and merge $\{ab\}$ and $\{ac\}$, which will be followed by the merging of $\{d\}$ and $\{e\}$. If we don't use the intermediate refinement, we merge $\{d\}$ and $\{e\}$ and right after we will merge $\{ab\}$ and $\{ac\}$.

In both cases we will create the clusters: $\{abc\}$ with $f(\{abc\}) = 2$ and $\{de\}$ with $f(\{de\}) = 1.2$. Next we will merge $\{abc\}$ with $\{d\}$ at $\mu(\{abc\}, \{d\}) = 1.3$. Here we have two clusters that are at a dissimilarity inferior than the f level of one of them ($\mu(\{abc\}, \{d\}) = 1.3 < f(\{abc\}) = 2$), if we use definitions (1) or (2) for the complete-link. That's why we use the extended indexing formula (A.1) when we compute the level of a new cluster with the complete-link defined as in (1) or (2). Using the complete-link defined as in (3) we will have the same result here: the creation of $\{abcd\}$ with $f(\{abcd\}) = 2$.

Thus, we will have two clusters that properly intersect themselves, $\{abcd\}$ and $\{de\}$, which will create the final cluster, $\{abcde\}$ with $f(\{abcde\}) = 3$, cf. Figure A.5.

Starting from the resulting hierarchy (cf. Figure A.3) and 2-3 hierarchy (cf. Figure A.5), we can compare the obtained ultrametric d_{AHC} from Figure A.6 and the induced dissimilarity matrix $d_{23\,AHC}$ for the 2-3 AHC from Figure A.7, in order to evaluate the quality of each method.

	b	c	d	e
a	1	3	1.4	1.4
b		3	1.4	1.4
c			3	3
d				1.2

Figure A.6: The resulting ultrametric

	b	c	d	e
a	1	1	2	3
b		2	2	3
c			2	3
d				1.2

Figure A.7: The resulting 2-3 ultrametric

As we can see, the two obtained distance matrices are different $d_{AHC} <> d_{23\,AHC}$, but when compared with the initial distance matrix we have: $d_{ini} < d_{AHC}$ and $d_{ini} <$

$d_{23\,AHC}$ as expected.

When we compare the obtained matrices with the initial data matrix, we will have a smaller deviation for the classical AHC method, than the 2-3 AHC method, or in other words a better quality for the AHC method compared with the 2-3 AHC method.

For example, using a simple difference between the resulting matrix and the initial one, we obtain for the AHC method a deviation $d_{AHC} - d_{ini} = 5.2$, compared with $d_{23\,AHC} - d_{ini} = 5.8$ for the 2-3 AHC method.

Appendix B

Single Link and "Normal" Execution of 2-3 AHC Algorithm

We present here a small data set (Figure B.1) and the created 2-3 hierarchy, using the single-link, the *double indexing formula* and the *normal* execution of the initial 2-3 AHC algorithm (without integrated refinement (**i**)). The created 2-3 hierarchy presents here a level inversion, which can be avoided using the integrated refinement (**ii**) or the *normal indexing formula* with a level test.

The data set is a set of five points with the dissimilarity matrix presented in Figure B.1. The 2-3 AHC algorithm will first merge the two singletons $\{a\}$ and $\{b\}$ found at minimum dissimilarity 1, and then it will merge $\{b\}$ and $\{c\}$ found at minimum dissimilarity 2. At this moment we will have two clusters that properly intersect themselves, $\{ab\}$ and $\{bc\}$, with $\mu(\{ab\}, \{bc\}) = 5$, Figure B.1.

	b	c	d	e
a	1	5	4	4
b		2	4	3
c			3	4
d				4

Figure B.1: The data set

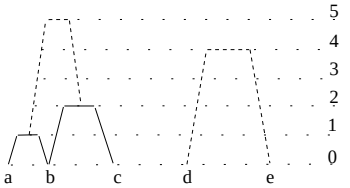


Figure B.2: Initial steps

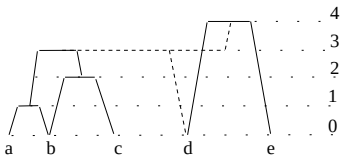


Figure B.3: Double indexing formula and the normal execution *(**i**)*

None of the clusters $\{a\}$, $\{b\}$, $\{c\}$, $\{ab\}$ and $\{bc\}$ can be merged with $\{d\}$ or $\{e\}$, and the only candidate pairs in this situation are $(\{ab\}, \{bc\})$ and $(\{d\}, \{e\})$, cf. Figure B.2. This is the 2-3 AHC algorithm execution so far, both for case (**i**) and case (**ii**) presented in Section 2.7.

If we use the *normal* algorithm execution, case (**i**), then the next merged clusters will be the singletons $\{d\}$ and $\{e\}$, since $\mu(\{d\}, \{e\}) = 4 < \mu(\{ab\}, \{bc\}) = 5$. After the merging of $\{d\}$ and $\{e\}$, the only pair of candidate clusters will be $(\{ab\}, \{bc\})$ which will be merged at dissimilarity 5. The level of $\{abc\}$ will be set to 3, using the double link indexing formula, since $\mu(\{abc\}, \{d\}) = \mu(\{abc\}, \{e\}) = \mu(\{abc\}, \{de\}) = 3$, cf. Figure B.3. Now, there are three pairs of candidate clusters: $(\{abc\}, \{d\})$, $(\{abc\}, \{e\})$ and $(\{abc\}, \{de\})$ with the same dissimilarity between candidates, $\mu = 3$, Figure B.3.

In order to have a richer 2-3 hierarchy, the smallest cardinality pairs are preferred, i.e. $(\{abc\}, \{d\})$ and $(\{abc\}, \{e\})$ and using also a lexicographical criterion the pair to be merged will be $(\{abc\}, \{d\})$; if we merge $(\{abc\}, \{e\})$ we will have the same result. After the merging of $\{d\}$ and $\{abc\}$ we will have the new cluster $\{abcd\}$ with $f(\{abcd\}) = 3$ and the only candidate pair will be $(\{abcd\}, \{de\})$, which properly intersects themselves. When we merge $\{abcd\}$ and $\{de\}$, the resulting cluster, $\{abcde\}$ will have $f(\{abcde\}) = 3$ using the double indexing formula, since $\mu(\{abcd\}, \{de\}) = 3$ and $\mu(\{abcde\}, T) = 0$ (there is no other cluster T). The successor $\{de\}$ of $\{abcde\}$ will have $f(\{de\}) = 4$, causing thus a level inversion in the created 2-3 hierarchy, cf. Figure B.4.

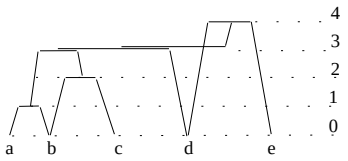


Figure B.4: Levels inversion for normal execution *(**i**)* of the 2-3 AHC algorithm

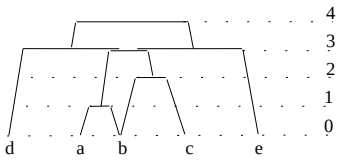


Figure B.5: 2-3 AHC algorithm with integrated refinement *(**ii**)*

This situation can be avoided if we use the *normal indexing formula* along with the level test in the normal execution or if we use directly the intermediate merging (**ii**).

In the case presented in Figure B.3, the *normal indexing formula* with the level test will reduce the level of the cluster $\{de\}$ to 3, when creating the final cluster $\{abcde\}$ during the normal algorithm execution (**i**).

Using the intermediate merging (**ii**), after the merging of $\{b\}$ and $\{c\}$ (Figure B.2) we will merge the two clusters that properly intersect each other, $\{ab\}$ and $\{bc\}$. The resulting cluster, $\{abc\}$, will have $f(\{abc\}) = 3$ (double indexing formula) and it will be a candidate cluster along $\{d\}$ and $\{e\}$ (Figure B.5). Here $\{d\}$ will not be merged any-

more with $\{e\}$, since $\mu(\{abc\}, \{d\}) = 3 < \mu(\{d\}, \{e\}) = 4$, instead $\{d\}$ will be merged with $\{abc\}$ and afterwards $\{e\}$ will also be merged with $\{abc\}$. We will have a proper intersection between $\{abcd\}$ and $\{abce\}$, with $f(\{abcd\}) = f(\{abce\}) = f(\{abc\}) = 3$ and $\mu(\{abcd\}, \{abce\}) = 4$, cf. Figure B.5. The final cluster $\{abcde\}$ will be thus created with $f(\{abcde\}) = 4$.

As we can see, the intermediate merging (**ii**) produce a richer 2-3 hierarchy and also avoids levels inversions caused by the "blocking" of clusters that properly intersect themselves.

Appendix C

Tests on Simulated Data

We present in this Appendix some results on the simulated datasets: the execution times in Section C.1 and the Stress gains in Section C.2.

C.1 Execution Times and Complexity

We begin with the execution times of the AHC and 2-3 AHC methods on the rectangle generated data (see Section 3.4.3 for more details). Since the execution times on the sinusoidal generated data are identical, we present below only the rectangle execution times.

The maximum execution times for the single link (SL) and the complete link (CL) are presented in figures C.1 and C.3. The maximum execution time complexity of the methods ($\frac{Execution\,time}{Number\,of\,elements}$) are presented in figure C.2 for the single link and in figure C.4 for the complete link. The corresponding average execution times and complexities are presented in figures D.5, D.6, D.7 and D.8.

As we can see the classical AHC algorithm and our 2-3 AHC algorithms perform in $\mathcal{O}(n^2 \log n)$ whilst the execution time of the initial 2-3 AHC algorithm explodes and its complexity curve does not remais constant. We confirm thus that the initial 2-3 AHC algorithm performs in $\mathcal{O}(n^3)$.

Note:
SL - Single Linkage
CL - Complete Linkage

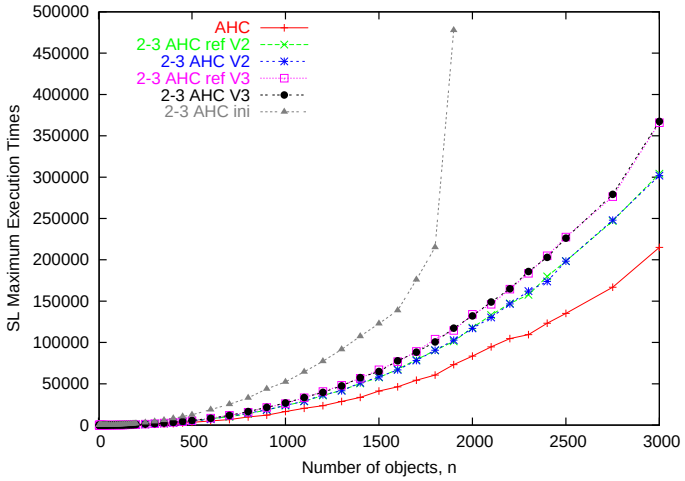


Figure C.1: SL maximum execution times

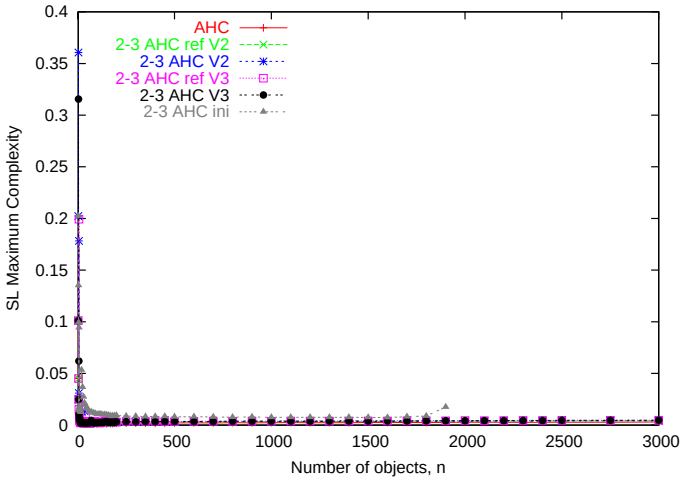


Figure C.2: SL maximum complexity

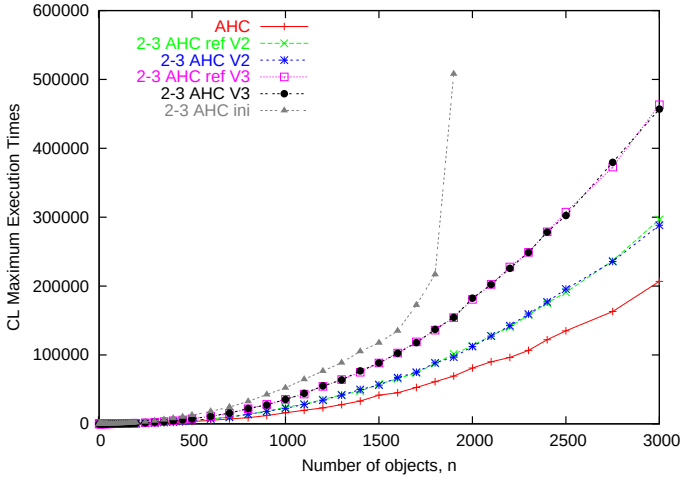


Figure C.3: CL maximum execution times

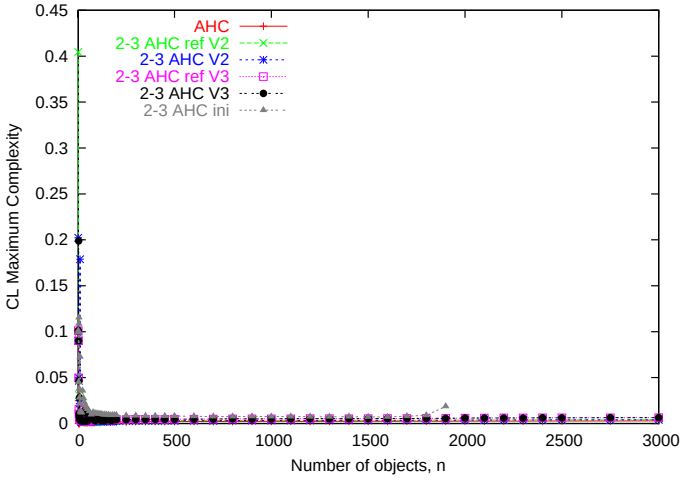


Figure C.4: CL maximum complexity

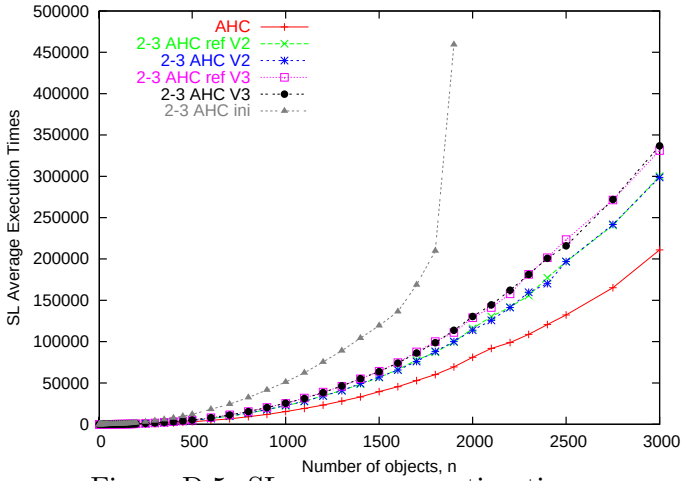


Figure D.5: SL average execution times

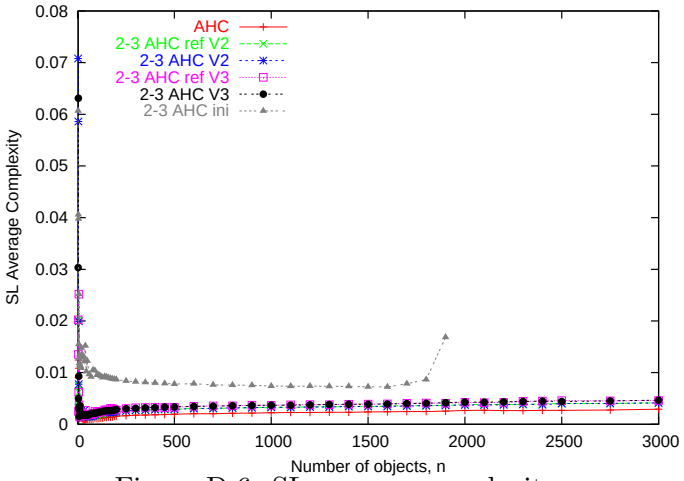


Figure D.6: SL average complexity

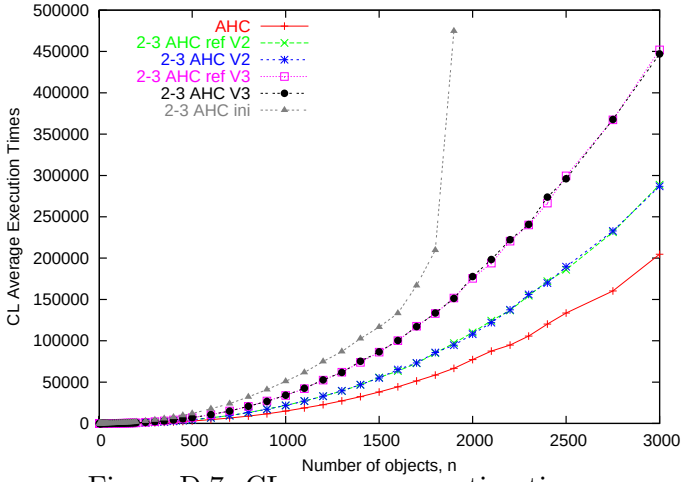


Figure D.7: CL average execution times

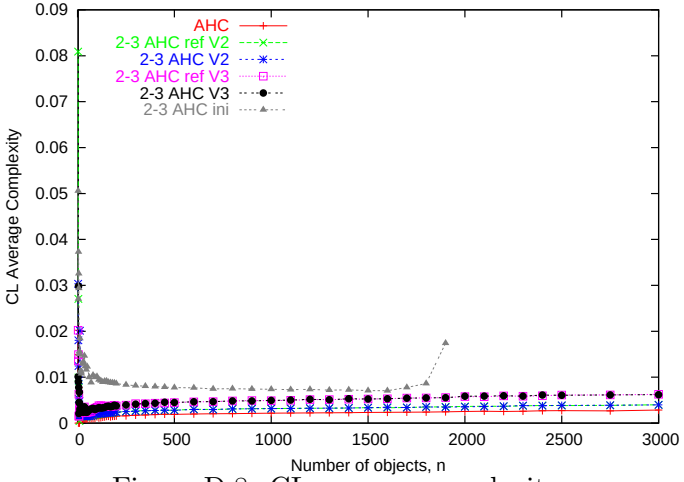
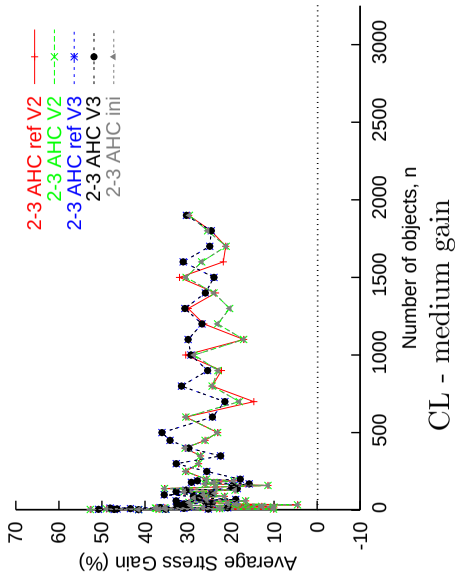


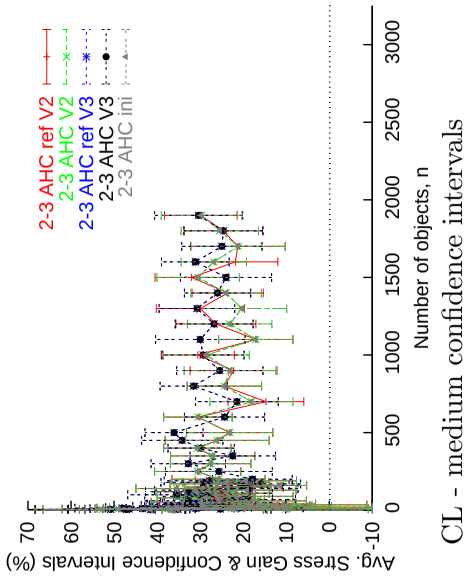
Figure D.8: CL average complexity

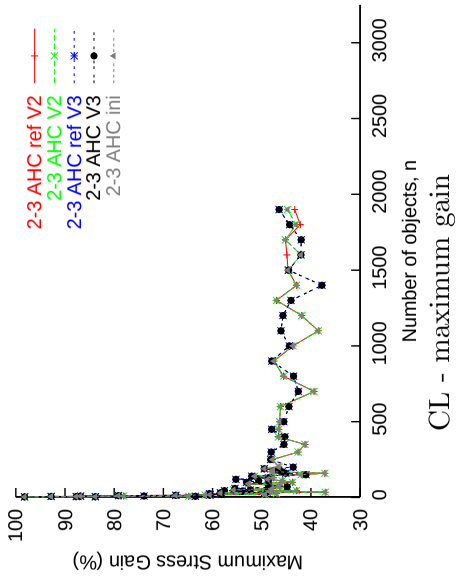
C.2 Stress Gain

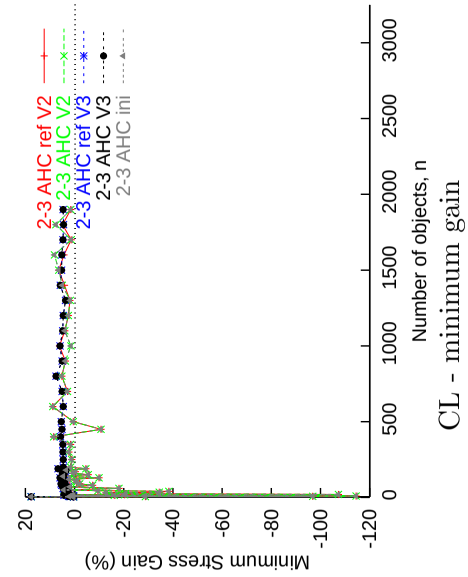
In this Section we present the Stress [JW82] gain obtained on the generated datasets. We begin with the rectangle generated data and continue with an example of the sinusoidal generated data.

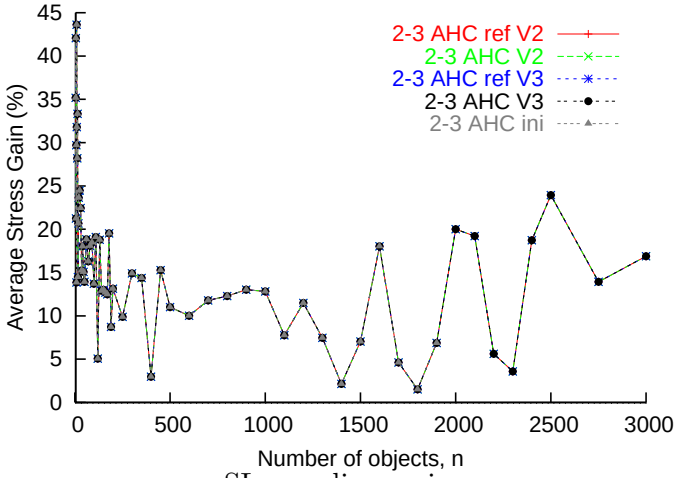
Rectangle Simulated Data



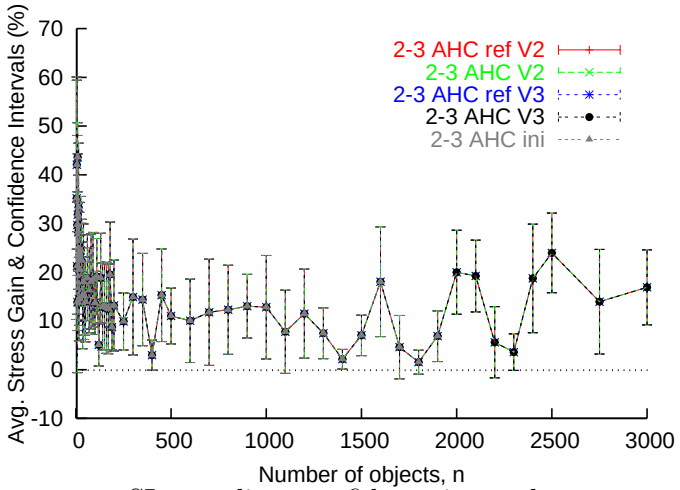




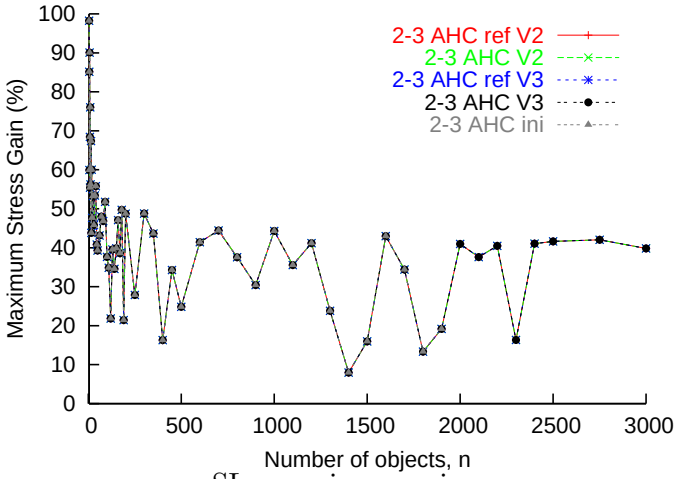




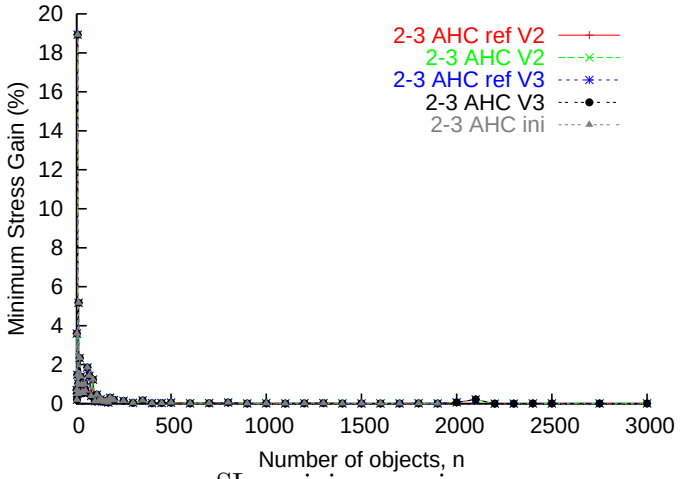
SL - medium gain



SL - medium confidence intervals



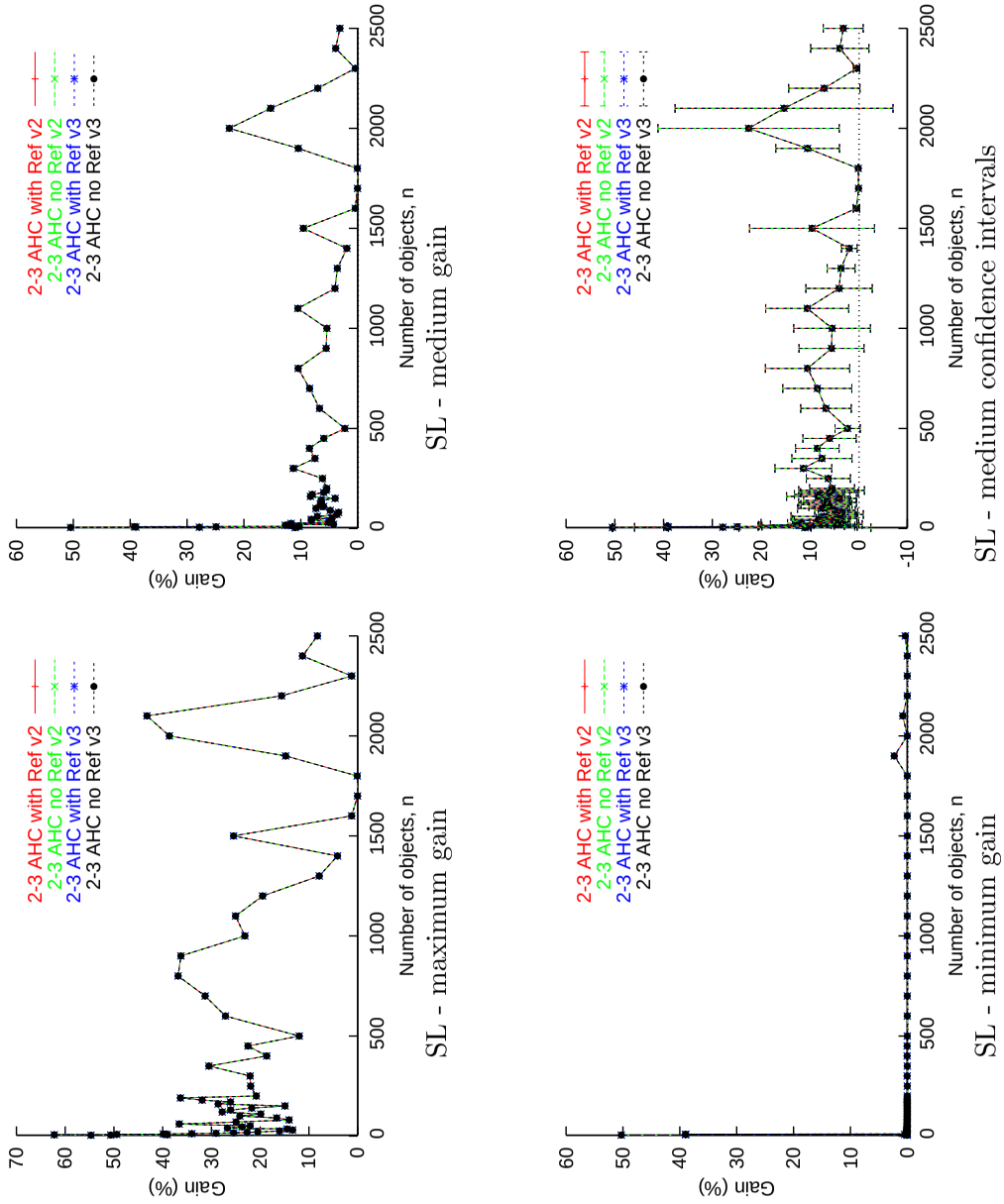
SL - maximum gain



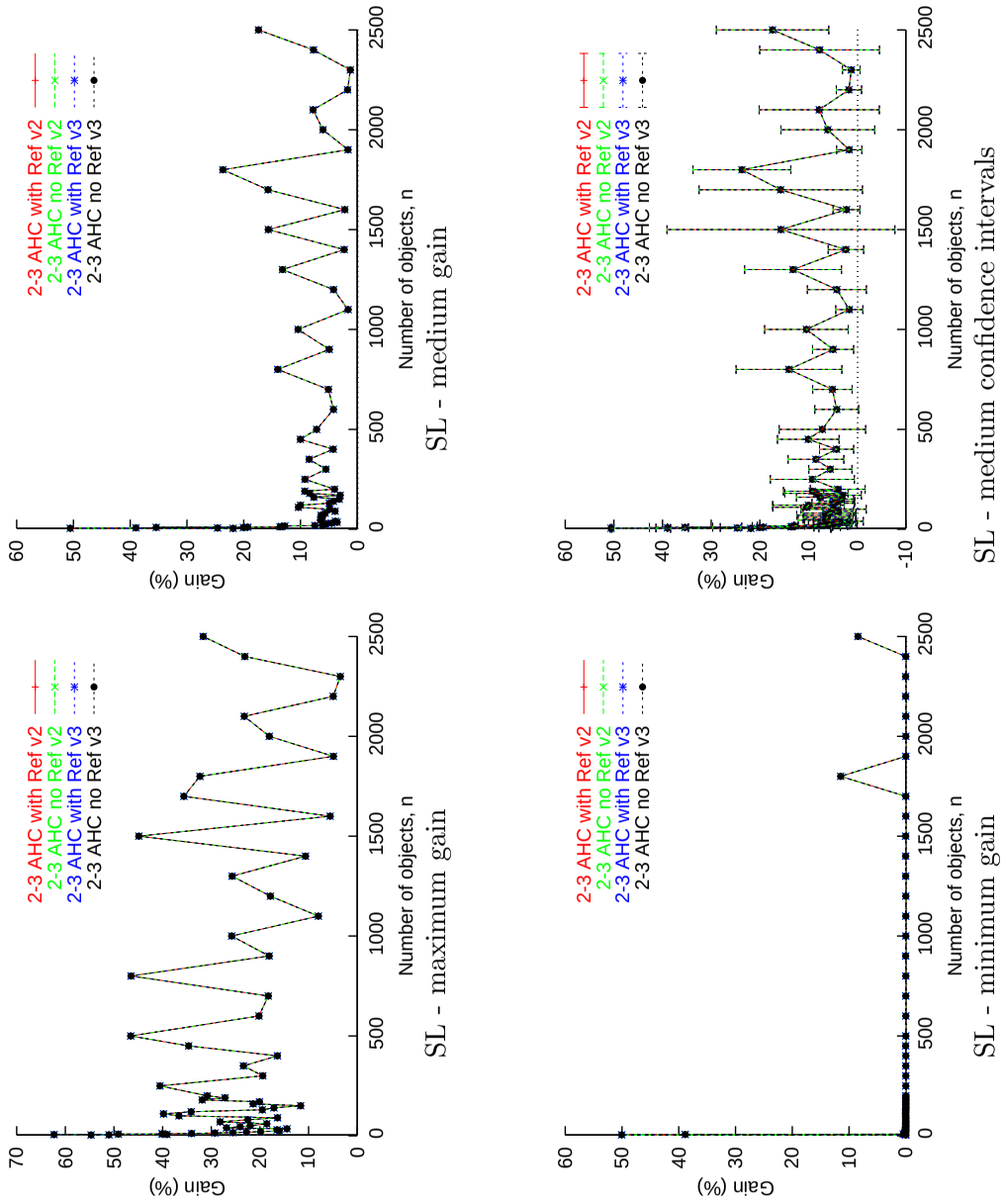
SL - minimum gain

Sinusoidal Data

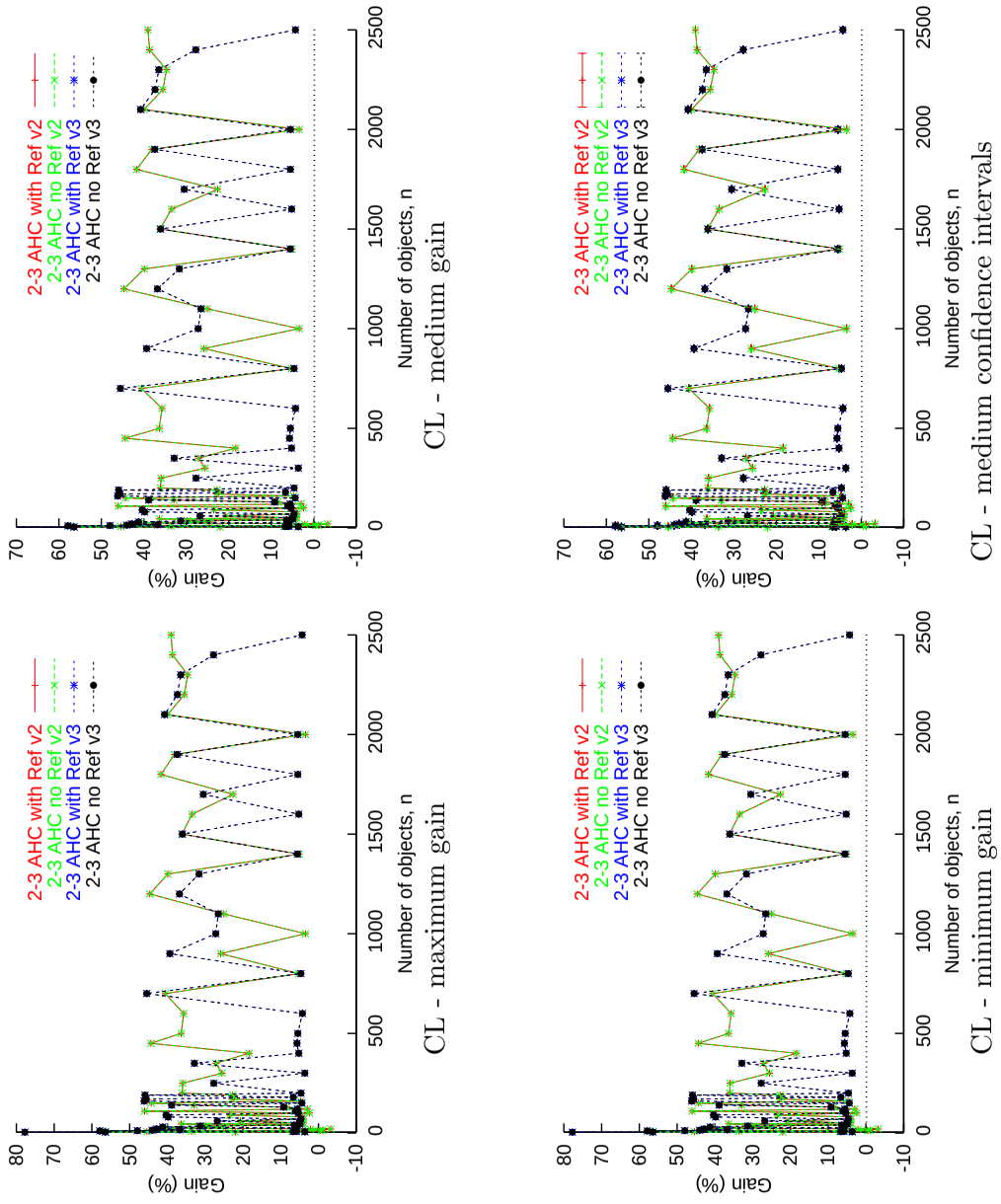
Single link; Noise z = 0



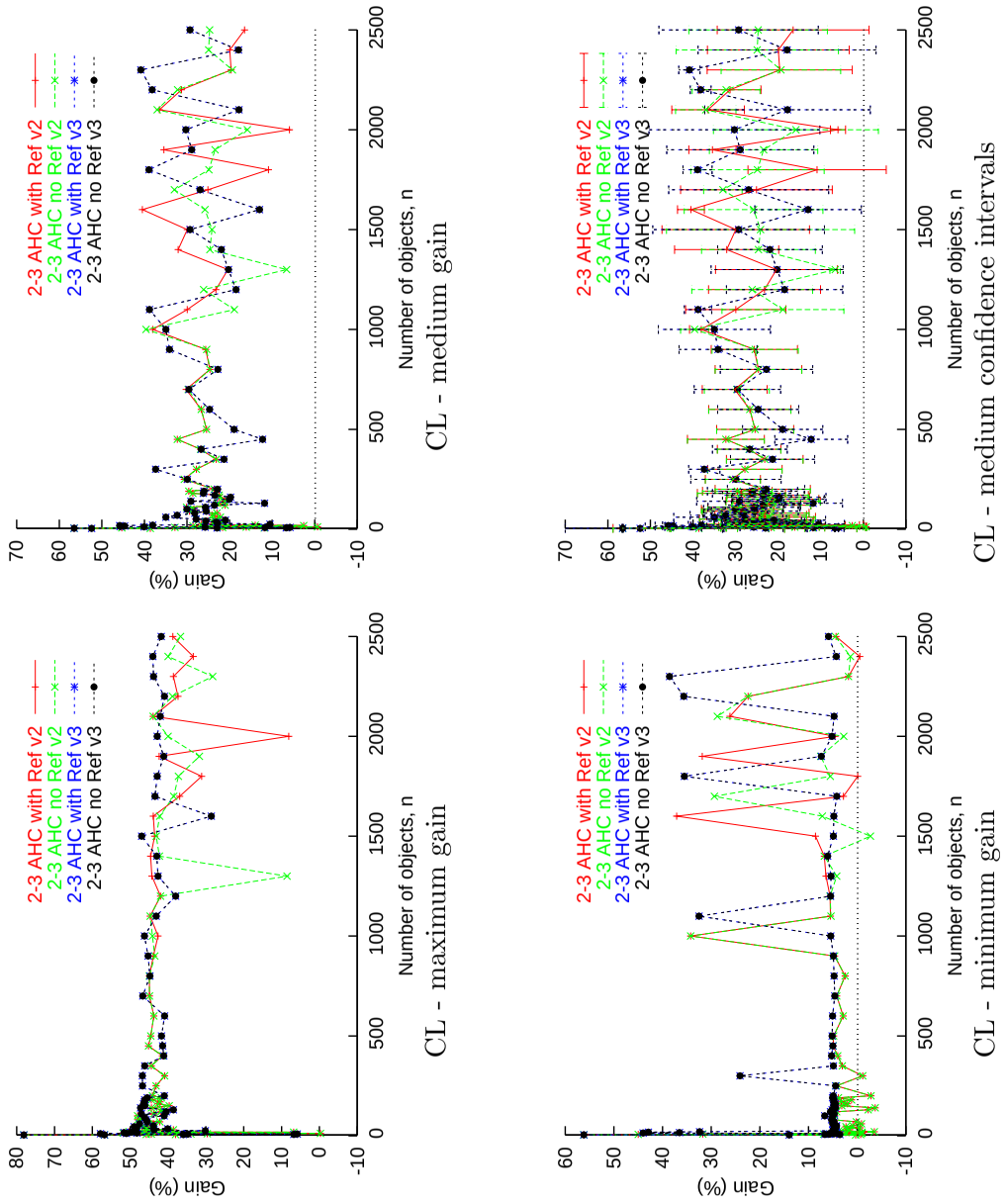
Single link; Noise z = 0.04



Complete link; Noise z = 0



Complete link; Noise z = 0.04



Appendix D

Use of HAC23Index as Part of the CBR*Tools in an CBR Application

To illustrate how our index can be initialized, respectively to point out its contribution to the retrieval phase's performance, we present shortly in the context of a application for car assurance risk factor determination, the steps for using different indexes of CBR*Tools. Implementing such an application based on the CBR*Tools suppose the instantiation of twelve classes from the framework. Details on the instantiation and use of our HAC23Index are given in this section. The classes for case attribute transformation and similarity calculus used by the neural index are used also by our index.

This application's objective is to determine the risk factor (taken over by the car assurance company) of a given car, based on previously established pairs of car parameters list and assigned risk values. In the CBR based approach, cases represent association between the car parameters description and the determined risk factor. For this application we use just two steps from the CBR cycle (cf Figure D.1), namely retrieve, reuse without revise and retain phases. The retrieve phase returns the set of cars which are the most similar with the newly presented car from the parameters viewpoint. Four search strategies are used during this phase:

- the k nearest neighbor algorithm;
- a tree structure-based algorithm, which use a pre-filtering process, realized through a hierarchy of prototypes (whereas by prototype we mean a group of similar cases/cars), and combined with the knn algorithm. This index is similar to our index, whereas clusters are equivalent to the prototypes used here;
- a neural index based algorithm, which basically in this context means a preprocessing phase of the case base, performing a "supervised" clustering of the cars

based on their descriptors and the assigned risk factor to them;

- the clusters structure-based algorithm combined with the knn algorithm.

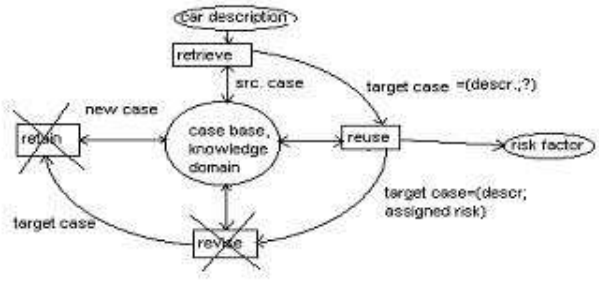


Figure D.1: CBR reasoning cycle in the CAR sample application

The reuse phase takes into account all of the returned cases by the previous step, and calculates a mean value of their risk factors using proportional weights with each ones similarity degree. Finally, the retain step adds the new case to the base, if the two following conditions hold: there is no already case stored in the memory which is very close to the new target case, and case base haven't reached its saturation level. We are interested here only in the first two phases of the CBR reasoning, this application having only a pedagogical/presentation purpose concerning these clustering indexes and the CBR paradigm.

D.1 Reasoning System Construction

The application's design can be decomposed in three phases based on the variability axes of the hot-spots of CBR*Tools, case representation, memory organization and reasoning maintenance axe. We will briefly discuss the case representation and the memory organization aspects, further technical details in [Jac98] and [Ben00].

D.2 Case Representation

Each item (car) from the database is characterized by twenty-five attributes of three possible types: integers, alphanumeric and floats.

```
public class CarCaseSituation extends JavaClassIndice {
    public int normalizedLosses;
    public String make;
    public float height;
    public int numOfCylinders;
...
```

```
}
```

From the CBR paradigm's standpoint, car parameters are of two types: potential item indices (all the attributes but the risk), and the solution assigned to the item (the risk factor). According to the framework definition, based on the case representation hot-spots (CbrCase and CompoundIndice), there are used two instantiations CarCase and CarCaseSituation [Ben00].

In the initialization phase of this class we can declare the indices, one indices for each attribute, defining their type(integer, float, symbol) and domain of values.

```
static private  int _cylinders []  = {2,3,4,5,6,8,12};
  static {
    ArrayIndiceDescriptor desc = new ArrayIndiceDescriptor(
        "aid.cbr.samples.cars.CarCaseSituation");

    desc.addIndiceDescriptor(new IndiceDescriptor("normalizedLosses",
          new RangeIntType(65,256)));
    desc.addIndiceDescriptor(new IndiceDescriptor("make",
          StringType.STRING));
    desc.addIndiceDescriptor(new IndiceDescriptor("height",
          new RangeFloatType(60,75)));
    desc.addIndiceDescriptor(new IndiceDescriptor("numOfCylinders",
          new ListIntType(_cylinders)) );
    ...
    }
```

D.3 Memory organization

Memory organization relies on three major hot-spots :

• case base (CaseBase) - specialized by the CarSimpleFileCaseBase class in this sample.

• similarity measure (Similarity) specialized by the CarSimilarity class, order (CmpValueOrder, CmpValue) relation-definition over the cases in the case base, and distance measure (DistanceMeasure) between the case specialized by PondEuclidDistance in this sample.

• index base (IndexBase) - which acts like a frame a frame for the different indexes defined over the current case base. Whereby index definition over a case-base, we mean the way of how a certain index algorithm makes use of and process the indices exposed by cases from that case-base through the CompoundIndice hot-spot.

D.4 Object oriented implementation

In the following we will focus on the similarity-order-distance and the index base axes of the sample application. Readers who are interested in a more exhaustive analyze of the case base hot-spot's specialization should point to [5].

The CarCaseSituation class contains all the twenty-five cars attributes as public members, and during a search phase an instance of this class will be passed to the Index hot-spot's search method. As a consequence, indexing classes have to transform in a way or in aase an instance of this class will be passed to the Index hot-spot's search method. As a consequence, indexing classes have to transform in a way or in annother the list of attributes in valid index values-relations, up to their need.

Our index uses three hot-spots, AttrTransform, ClusterDistance and DistanceMeasure. The former one ensures the mapping of car attributes from the CarCaseSituation in valid scalar values, whereupon it can be applied different norms or distance metrics subsequently. Distance metrics are specializations of the DistanceMeasure hot-spot and the cluster distances (*single-linkage, complete-linkage* etc.) are specializations of the ClusterDistance hot-spot and make use of the DistanceMeasure hot-spot for their singletons.

```
public class BasicAttrTransform implements AttrsTransform{
  public double[] compIndiceTransform(CompoundIndice attrs){
    int nrOfAttrs=attrs.size();
    double numericAttr[]=new double[nrOfAttrs];
    IndiceType indType;
    ...
    Object transfList = _attrTransfInfo[ind]._transfList,
         attr=null,
         attrVal;
    ...
         if(attrVal instanceof Float){
              numericAttr[ind] = ((Float)attrVal).floatValue();
         }
         else if(attrVal instanceof Integer){
    numericAttr[ind] = ((Integer)attrVal).intValue();
  }
              else{
        numericAttr[ind]-= _attrTransfInfo[ind]._minVal;
                numericAttr[ind]/= _attrTransfInfo[ind]._maxVal-
                                   _attrTransfInfo[ind]._minVal;
 }
return numericAttr;
```

```
  }

public class SingleLinkageDistance implements ClusterDistance{
  private DistanceMeasure _singletonsDistance;
  ...
  public Float getDistance(ClusterPrototype newCluster,
      ClusterPrototype otherCluster, SortedMap clusterTree){
  ...
    if(newCluster._succ.isEmpty() && otherCluster._succ.isEmpty())
       return new Float(getDistance(newCluster._refVector,
                                    otherCluster._refVector));
    else{
         ...
         return (Float)clusterTree.get(toKeyPair(otherCluster._id,
                                                 otherCluster._id));
        }
  }
```

The CarIndexBase (IndexBase hot-spot) serves as a frame for the defined indexes and index strategies, whereby strategy we mean a sequential apply of two or more indexes. Each index and indexing strategy, once defined, have to be registered to it in order to became accessible for the Reasoner hot-spot. Having defined and registered all the indexes and strategies one might want to use during the retrieve phase, one can dynamically activate the selected strategy through the IndexParams hot-spot of the Reasoner's RetrieveStep hot-spot [Jac98].

The results obtained with our HAC23Index were similar to the ones obtain by the neural index, but implied the apriori partitioning treshold specification. Currently automatic criteria for choosing the indexing level partitioning are under study for future implementations along with a possible incremental feature for an on-line analysis.

Appendix E

INRIA Research Teams Organization

Before 1^{st} of April 2004, INRIA's research teams were organized in four different research themes, namely:

- Theme **1**: Networks and systems:
  - **A** : Architectures and Systems,
  - **B** : Networks and Telecommunications,
  - **C** : Distributed and Real-Time Programming.
- Theme **2**: Software engineering and symbolic computing:
  - **A** : Semantics and Programming,
  - **B** : Algorithms and Computational Algebra.
- Theme **3**: Human-computer interaction, images processing, data management, knowledge systems:
  - **A** : Databases, Knowledge Bases, Cognitive Systems,
  - **B** : Vision, Image Analysis and Synthesis.
- Theme **4**: Simulation and optimization of complex systems:
  - **A** : Control, Robotics, Signal,
  - **B** : Modelling and Scientific Computing.

After this date, the research teams were reorganized in the following five research themes (see also `http://www.inria.fr/recherche/equipes/listes/index.en.html`):

- Theme **Com**: Communicating systems:
  - **A** : Distributed systems and software architecture,
  - **B** : Networks and telecoms,
  - **C** : Embedded systems and mobility,
  - **D** : Architecture and compiling.
- Theme **Cog**: Cognitive systems:

- **A** : Statistical modeling and machine learning,
- **B** : Perception, indexing and communication for images and video,
- **C** : Multimedia data: interpretation and man-machine interaction,
- **D** : Image synthesis and virtual reality.

- Theme **Sym**: Symbolic systems:
  - **A** : Reliability and safety of software,
  - **B** : Algebraic and geometric structures, algorithms,
  - **C** : Management and processing of language and data.
- Theme **Num**: Numerical systems:
  - **A** : Control and complex systems,
  - **B** : Grids and high-performance computing,
  - **C** : Optimization and inverse problems for stochastic or large-scale systems,
  - **D** : Modeling, simulation and numerical analysis.
- Theme **Bio**: Biological systems:
  - **A** : Modeling and simulation in biology and medicine.

The Web pages on the national server that were influenced by the reorganization of the research themes are presented in Table E.1 below.

http://www.inria.fr/recherche/equipes/listes/index.fr.html
http://www.inria.fr/recherche/equipes/listes/index.en.html
http://www.inria.fr/recherche/equipes/listes/theme_1.en.html
http://www.inria.fr/recherche/equipes/listes/theme_1.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_2.en.html
http://www.inria.fr/recherche/equipes/listes/theme_2.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_3.en.html
http://www.inria.fr/recherche/equipes/listes/theme_3.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_4.en.html
http://www.inria.fr/recherche/equipes/listes/theme_4.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_Bio.en.html
http://www.inria.fr/recherche/equipes/listes/theme_Bio.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_Cog.en.html
http://www.inria.fr/recherche/equipes/listes/theme_Cog.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_Com.en.html
http://www.inria.fr/recherche/equipes/listes/theme_Com.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_Num.en.html
http://www.inria.fr/recherche/equipes/listes/theme_Num.fr.html
http://www.inria.fr/recherche/equipes/listes/theme_Sym.en.html
http://www.inria.fr/recherche/equipes/listes/theme_Sym.fr.html

Table E.1: National Web pages presenting the research themes organization

Team	Old Theme	New Theme
AXIS	3A	CogA
ORPAILLEUR	3A	CogA
MAIA	3A	CogA
DREAM	3A	CogA
CORTEX	3A	CogA
CORDIAL	3A	CogC
PAROLE	3A	CogC
ORION	3A	CogC
METISS	3A	CogC
MERLIN	3A	CogC
I3D	3A	CogC
ECOO	3A	CogC
IN-SITU	3A	CogC
LANGUE ET DIALOGUE	3A	SymC
ACACIA	3A	SymC
ATOLL	3A	SymC
EXMO	3A	SymC
TEXMEX	3A	SymC
SYMBIOSE	3A	BioA
HELIX	3A	BioA
EIFFEL	3A	-
OPERA	3A	-
VERSO	3A	-
CARAVEL	3A	-
AID	3A	-
IMEDIA	3B	CogB
MOVI	3B	CogB
TEMICS	3B	CogB
VISTA	3B	CogB
ARIANA	3B	CogB
ISA	3B	CogD
ALCOVE	3B	CogD
MIRAGES	3B	CogD
SIAMES	3B	CogD
REVES	3B	CogD
ODYSSEE	3B	BioA
EPIDAURE	3B	BioA
IMAGIS	3B	-
PASTIS	3B	-
SHARP	3B	-
AIR	3B	-

Table E.2: Teams from theme 3 in the old and new research themes

Appendix F

Be-TRIP Recommender System

This Appendix contains a publication (in French) on the specifications of the Be-TRIP recommender system. This work was published in proceedings of the 1st French-speaking conference on Mobility and Ubiquity computing [CGT04].

Recommandations personnalisées pour la recherche d'information facilitant les déplacements

Sergiu Chelcea
AxIS, INRIA Sophia Antipolis
2004 Rte des Lucioles, BP 93
06902 Sophia Antipolis Cedex, France
schelcea@sophia.inria.fr

George Gallais
VISA, INRIA Sophia Antipolis
2004 Rte des Lucioles, BP 93
06902 Sophia Antipolis Cedex, France
ggallais@sophia.inria.fr

Brigitte Trousse
AxIS, INRIA Sophia Antipolis
2004 Rte des Lucioles, BP 93
06902 Sophia Antipolis Cedex, France
trousse@sophia.inria.fr

ABSTRACT

This article concerns an emerging research field related to mobility from the transport point of view, which is the information search for traveling/mobility. To facilitate such a search, we propose the use of recommender systems in a mobility context: these facilitate on the one hand the information search, and on the other hand these help to prepare the user's trip ("pre-trip": choice of the transport mode, schedule, route, time of the trip, ...) and to carry it out ("on-trip": interactive guidance, way visualization, destination planning). This double impact is rarely exploited today and we propose, after a description of the used technologies, to illustrate the potentials of this new approach on a traditional tourist visit example. The originality of this approach lies in 1) its capacities to adapt the recommendations to the user's behavior during his information retrieval correlated to his own movement and 2) the on-line learning capacities of such a system for information search assistance.

Cet article aborde un domaine de recherche en plein essor relatif à la mobilité sous l'angle du transport en y associant la recherche d'information pour se déplacer. Pour faciliter une telle recherche, nous proposons l'utilisation de systèmes de recommandations dans un contexte de mobilité qui ont un double impact : faciliter d'une part la recherche d'information, mais aussi, aider à préparer le déplacement ("pre-trip" : choix du mode de transport, horaire, itinéraire, temps de trajet, ...) et à le réaliser ("on-trip" : guidage interactif, aide à un changement de destination, visualisation trajet). Ce double impact est aujourd'hui rarement exploité et nous proposons, après une description des technologies sous-jacentes, d'illustrer par un exemple classique de visite touristique les potentiels de cette nouvelle approche. L'originalité de cette approche réside dans 1) ses capacités d'adaptation des recommandations au comportement de l'utilisateur lors de sa recherche d'information corrélé à son déplacement effectif et 2) les capacités d'apprentissage en-ligne d'un tel système d'aide à la recherche d'information.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, clustering*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*User-Centered Design*

General Terms

Information Retrieval, Adaptive Interfaces, Mobility

Keywords

Systèmes de recommandations, filtrage collaboratif, raisonnement à partir de cas, information voyageur, tourisme, recommender systems, collaborative filtering, case-based reasoning, traveller information, pre-trip, on-trip, tourism, transport

1. INTRODUCTION

Le développement rapide des systèmes d'information nomades permet d'envisager de nombreuses applications dans un contexte de mobilité. Cependant, cette mobilité prise sous l'angle du transport, est aussi un domaine de recherche en plein essor lorsque on y associe, dans une optique de sauvegarde de l'environnement, la recherche d'information pour un déplacement "optimum". Les acteurs du domaine des ITS - Intelligent Transport Systems - décrivent souvent l'apport des technologies de l'information à travers des scénarios de déplacement. Il n'est pas dans notre intention d'en ajouter un mais d'illustrer notre approche à travers une chaîne de déplacement dite de porte à porte.

Le déplacement est toujours la conséquence d'une information (un rendez vous, une manifestation culturelle, ...). Prenons l'exemple d'un internaute découvrant une manifestation culturelle au musée Picasso d'Antibes à quelques kilomètres de son domicile (pre-trip) ou de sa position actuelle lors d'un déplacement (on-trip) et à laquelle il décide de se rendre spontanément. Celui-ci va avoir besoin de rechercher les heures d'ouverture, l'adresse et comment s'y rendre. D'une part les informations qui lui seront nécessaires sont éparpillées, d'autre part le contexte de son déplacement (individuel, en groupe, ...) et les critères de choix du mode de déplacement (confort, temps, coût) sont par essence de sa propre initiative. Enfin il devra pouvoir les consulter tout au long de sa chaîne de déplacement, quelque soit son terminal et en fonction du lieu. Il est clair que les PDA communiquant vont permettre d'assurer une continuité d'accès à l'informa-

tion tout au long du déplacement mais au prix d'une masse de données consultable d'autant plus importante.

Dans ce contexte, les systèmes de recommandations ont un double impact : aider à la préparation du déplacement ("pre-trip" : choix du mode de transport, horaire, itinéraire, temps de trajet, etc.) et assister pendant le déplacement ("on-trip" : guidage interactif dans le cas d'un transport individuel, visualisation du trajet dans le cas d'un transport collectif).

Ce double impact est aujourd'hui rarement exploité et nous proposons, après une description des technologies sous-jacentes en section 2 issues de l'intelligence artificielle, d'illustrer en section 3 par un premier exemple de visite culturelle les potentialités de l'approche Broadway de calcul de recommandations personnalisées. celle-ci est basée sur la similarité de comportements utilisateurs d'une part dans leur recherche d'informations coorélés à leur déplacement effectif. Une spécification d'un système de recommandations appelé Be-TRIP basée sur cette approche est donnée en section 3, visant une aide à la fois au déplacement pre-trip et on-trip. En section 4, nous nous comparons aux principaux travaux similaires pour conclure sur nos travaux en cours et futurs.

2. RECOMMANDATIONS EN MOBILITÉ

Après avoir introduit la notion de systèmes de recommandations, nous indiquons en quoi de tels systèmes trouvent leur intérêt dans un contexte de mobilité, permettant de trier cette masse considérable d'information accessible à l'internaute.

2.1 Systèmes de recommandations

L'objectif d'un système de recommandations est d'aider les utilisateurs à faire leurs choix dans un domaine où ils disposent de peu d'informations pour trier et évaluer les alternatives possibles. Un système de recommandations [10] peut être décomposé en trois entités de base (cf. Figure 1) : le groupe d'agents *producteurs* de recommandations, le module de *calcul de recommandations* et le groupe de *consommateurs* des recommandations. Un défi majeur dans le domaine de la conception de systèmes de recommandations est le suivant :

> Comment produire des recommandations *personnalisées* et de haute *qualité* tout en *minimisant l'effort* requis de la part des producteurs et des consommateurs?

Deux grandes approches [10] complémentaires sont proposées dans la littérature ainsi que des approches hybrides :

1) l'approche (notée A.P.U dans le tableau 1) basée sur le contenu et fondée sur l'apprentissage automatique de profils utilisateurs. Les recommandations sont issues uniquement des actions passées de l'utilisateur lui-même.
2) l'approche (notée F.C. dans le tableau 1) dite de *filtrage collaboratif* où des données issues d'autres utilisateurs sont utilisées dans le calcul de recommandation (par exemple des recommandations, des sessions utilisateurs, etc.). Généralement cette approche est fondée sur des techniques de fouille de données.

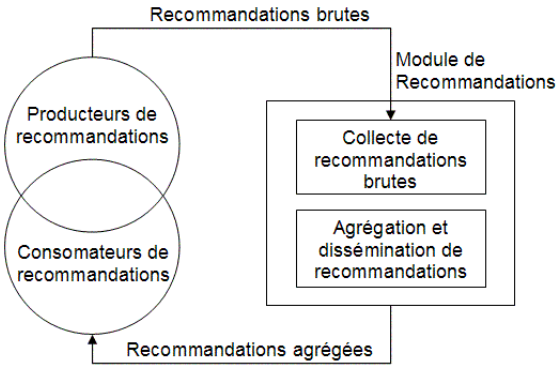


Fig. 1 – ***Architecture générale d'un système de recommandation***

Le profil utilisateur est une structure de données qui décrit en particulier les centres d'intérêts d'un utilisateur dans l'espace des objets à recommander. Celui-ci est une structure construite dans la première approche ou donnée dans la seconde par l'utilisateur. Ce profil est utilisé soit pour filtrer les objets disponibles (on parle alors de filtrage basé sur le contenu), soit pour recommander à l'utilisateur ceux qui ont satisfait d'autres utilisateurs ayant un profil similaire (on parle alors de filtrage collaboratif) [10]. Il est à noter l'existence d'approches hybrides comme le permet notre approche (Cf. section 3.) basant le calcul de recommandations sur des similarités de sessions ou/et de profils qui peuvent être appris ou donnés. Notons enfin que le profil utilisateur peut comprendre également des informations sur les dispositifs utilisés et préférences utilisateur en termes de services pour rendre plus aisée l'adaptation des interfaces à l'utilisateur.

2.2 Intérêt de tels systèmes d'aide en mobilité

On sait aussi que du côté de la demande, les consommateurs cherchent des produits (de tourisme par exemple) personnalisés et demandent d'accéder à l'information appropriée et à des services de haute qualité sur le Web, à tout moment et n'importe où dans un environnement mobile. Or le Web est un gigantesque hypermédia dans lequel trouver un document pertinent n'est pas une tâche facile malgré les outils existants pour effectuer une recherche.
En contexte de mobilité, la localisation géographique de l'utilisateur est le premier filtre qui va venir s'ajouter au traditionnel filtre du profil de l'utilisateur. Les premiers services offerts par la localisation ont été développés dans les systèmes de navigation pour automobile maintenant largement diffusés. Le conducteur peut ainsi recevoir des conseils de guidage tout au long de son trajet, avec des messages visuels ou audio lui indiquant la route à suivre. Ces conseils utilisent la connaissance de l'itinéraire préalablement calculé et de la position "métrique" instantanée du véhicule. Plus récemment, les "Location-based Services ou LBS", utilisant la position du mobile obtenue par triangulation permettent d'envisager le même type de services pour des piétons.

Dans notre scénario, la découverte de l'information qui déclenche le besoin de mobilité se fera à partir d'un type de terminal et selon une session sur Internet qui permettront de

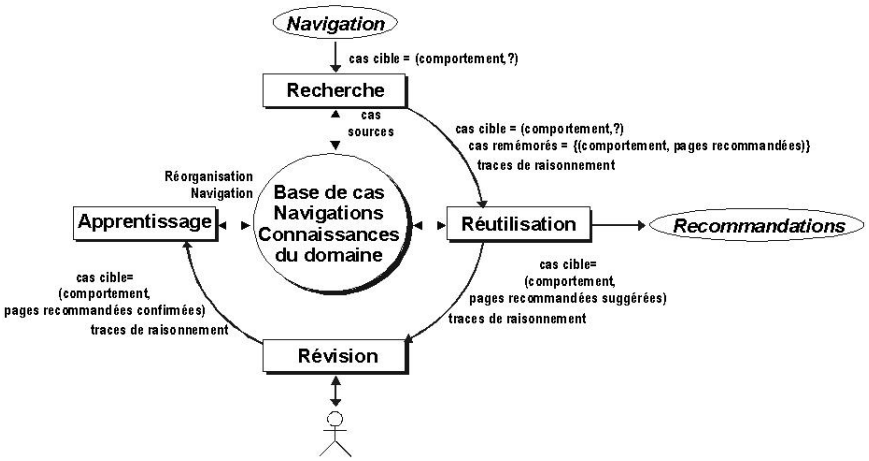


Fig. 2 – *Calcul de recommandations avec l'approche Broadway*

différencier les phases de préparation ou de déplacement. Dans la phase de préparation du voyage, le terminal sera soit de type PC sur réseau filaire soit la position de son terminal sera fixé. Ensuite, une fois le choix du mode de déplacement effectué, les caractéristiques du terminal et de réseau auront de fortes chances d'évoluer, et en associant de nouveau la position dynamique et l'itinéraire choisi puis stocké dans le terminal portable, de nouveaux services vont pouvoir être offerts, bien au-delà du guidage décrit précédemment car ceux-ci peuvent être généralisés à tout mode de déplacements (individuel ou collectif). On pourra par exemple, dans le cas du choix d'un transport en commun, en déduire que l'utilisateur est disponible, car il n'a pas à conduire, et lui délivrer des informations plus complètes personnalisées sur le trajet (temps de trajet restant, prochain arrêt, site touristique le long du trajet, accès courrier électronique etc.. .), valorisant ainsi le transport en commun. On pourra également lui suggérant des nouvelles intentions de déplacement en fonction de son déplacement réel et de sa recherche sur le web.

Par ailleurs, les recommandations personnalisées s'appliquent pour restituer l'information en s'adaptant automatiquement au terminal (taille et résolution d'écran), à son mode d'accès (clavier de bureau, stylet des PDA, reconnaissance vocale), à sa puissance de calcul, ses capacités de stockage et bien sur les capacités du réseau de communication. L'intérêt pour les personnes à mobilité réduite est évident : passage automatique du terminal en mode vocale, sélection transparente des systèmes d'information adaptés aux déficients visuels, choix des modes de transport adaptés (transport en commun, transport à la demande), délivrance d'un message personnalisé à l'arrivée.

Un deuxième niveau d'impact de l'utilisation de tels systèmes de recommandations est pour le concepteur du système d'information. En effet l'analyse des traces du recommandeur et en particulier de la qualité des recommandations peuvent donner lieu à des restructurations du système d'information.

3. BE-TRIP, AIDE AU DÉPLACEMENT

Cette section introduit notre approche de calcul de recommandations personnalisées appelée l'approche BROADWAY [1] basée sur des techniques de *Raisonnement à Partir de Cas* (RàPC). Puis elle présente la conception du système Be-TRIP (*Broadway-Extended pre and on-TRIP assistant*) d'aide au déplacement. Une illustration, après quelques éléments d'implémentation est donnée pour un scénario touristique en pre-trip.

3.1 L'approche Broadway

L'approche Broadway [11] est une approche hybride de calcul de recommandations pour l'aide à la recherche d'informations : celle-ci est à la fois basée sur l'analyse du contenu visité et centrée fouille de données où les comportements passés d'un groupe d'utilisateurs sont utilisés pour calculer les recommandations (cf. filtrage collaboratif [10]). L'originalité de l'approche Broadway, à la différence de la plupart des autres approches, réside dans la prise en compte de l'ordre des actions des utilisateurs comme un élément central dans la calcul de recommandations. En effet la plupart des approches fondées sur la fouille de données sont principalement des approches statistiques où l'ordre d'occurrence d'événements. dans l'historique n'est pas pris en compte lors du calcul de recommandations. Citons comme exemple, dans le domaine d'aide à la navigation sur le Web, le système FootPrints [12] et le système de Yan et al [13].

De manière générale, la réutilisation de comportements utilisateur dans l'approche Broadway s'appuie sur le *Raisonnement à Partir de Cas* (RàPC) [1] qui est une approche de résolution de problèmes basée sur la réutilisation par analogie d'expériences passées appelées "cas". Le RàPC se décompose habituellement en quatre phases principales [1] (cf. Figure 2) : recherche de cas, réutilisation, révision de la solution proposée, apprentissage. Enfin un cas est généralement indexé pour permettre de le retrouver suivant certaines caractéristiques pertinentes et discriminantes, appelées "indices"; ces indices déterminent dans quelle situation (ou contexte) un cas peut être de nouveau réutilisé.

1. `Broadway` – BROwsing ADvisor reusing pathWAYs, http://www-sop.inria.fr/axis/broadway/

La mise en oeuvre de l'approche Broadway nécessite la manipulation des cas dont les indices sont constitués de séquences d'événements. Pour cela, nous nous appuyons sur le *modèle d'indexation par situations comportementales* développé dans le cadre de le thèse de Jaczynski [8]. Pour l'utilisation de ce modèle, nous avons identifié quatre étapes importantes : 1) identification des variables d'observation, b) détermination de la sémantique d'un enregistrement (navigation) et de son contexte, c) définition de la représentation des cas et des *situations comportementales* (éléments issus d'une navigation pour caractériser un cas) et enfin d) conception des phases de raisonnement.

Les cas sont issus de l'instanciation d'un patron de cas potentiels. Ce patron repose sur des règles de construction d'une *situation comportementale* et d'une liste d'actions à recommander (par exemple une liste de pages évaluées) à un instant donné à l'intérieur d'une navigation. Selon ces règles, la *situation comportementale* est alors composée d'une composante instantanée contenant le contexte de la navigation (par exemple, noms des sites visités et mots clefs extraits des pages visitées) et d'une composante comportementale référençant un instant précis dans la navigation.

Le calcul des recommandations est alors effectué durant un raisonnement en quatre phases (cf. Figure 2), étant donnée la navigation courante d'un utilisateur :

1. Recherche. La base de cas est parcourue pour identifier les cas dont la partie comportement peut être mis en correspondance avec la navigation courante. Les meilleurs cas sont alors retournés.
2. Réutilisation. Chaque cas retrouvé propose une liste de pages recommandées, et la phase de réutilisation construit une liste ordonnée de ces pages à l'aide de critères divers afin de les suggérer à l'utilisateur courant.
3. Révision. La phase de révision est effectuée par l'utilisateur lorsqu'il continue sa navigation et qu'il évalue les pages visitées s'il le désire. La révision se poursuit jusqu'à la fermeture de la navigation courante et met en attente tous les raisonnements effectués pour une navigation (pages recommandées confirmées).
4. Apprentissage et maintenance de la mémoire. L'apprentissage prend en compte la navigation courante et l'ensemble des raisonnements associés pour en faire leur synthèse. La mémoire est alors mise à jour par la création de nouveaux cas, par la mise à jour des cas et par l'ajout de la navigation courante. A côté de ce processus de raisonnement, les cas et les navigations devenus obsolètes sont régulièrement effacés de la mémoire.

L'originalité de notre approche en filtrage collaboratif réside dans 1) ses capacités d'adaptation des recommandations au comportement de l'utilisateur observé selon diverses variables (cf. Figure 3) voire à son profil (centres d'intérêt, dispositif utilisé) et 2) les capacités d'apprentissage en-ligne d'un tel système d'aide à la recherche d'information.

Deux principaux assistants d'aide à la navigation basés sur l'approche Broadway ont été réalisés : Broadway-Web (1997) pour l'aide à la navigation sur le Web et Broadway-AT (2000) utilisé dans le cadre de l'application Educaid (France Telecom Inria) pour l'aide à la navigation dans un portail de ressources éducatives. Une application basée sur une adaptation de Broadway-AT a permis des recommandations personnalisées basées sur la similarité entre comportements non visuels et visuels des utilisateurs d'un site contenant les rapports d'activités des équipes Inria.

Nous spécifions ci-après les principales caractéristiques de l'assistant d'aide personnalisée au déplacement (pre-trip et on-trip) Be-TRIP ainsi que son intégration dans un serveur d'applications en cours de conception à l'Inria.

3.2 Spécifications de Be-TRIP

La spécification du recommandeur Be-TRIP consiste en une adaptation d'un premier système appelé Broadway-Web pour l'aide à la navigation sur le web au contexte de la mobilité et appliqué à l'aide au déplacement.

Be-TRIP est basée sur l'hypothèse suivante (cf. Figure 3): *si deux utilisateurs (ayant éventuellement des profils similaires) ont navigué suivant une séquence similaire de pages Web (ou actions Web) et/ou selon un chemin similaire (aspect localisation), c'est qu'ils ont des intérêts similaires, on peut alors proposer au second utilisateur les pages (ou actions) jugées pertinentes*[2] *par le premier* (Figure 3).

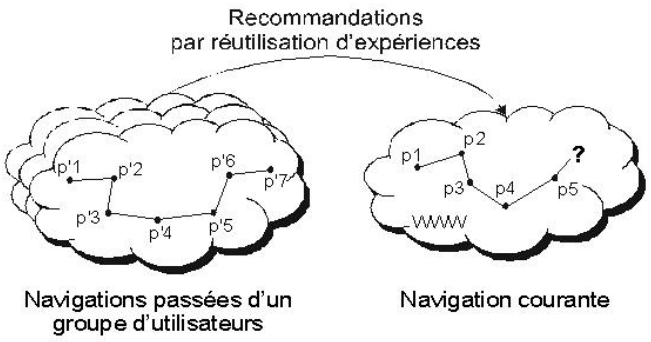


Fig. 3 – ***Réutilisation d'expériences pour déterminer des recommandations***

Le raisonnement dans Broadway-Web s'appuie sur l'application de notre modèle d'indexation par situation comportementale basé sur les quatre étapes suivantes déjà présentés :

Identification des variables d'observation. L'observation du comportement des utilisateurs est fondée sur l'utilisation

- de quatre variables relatives à leur recherche d'informations sur le Web pour le déplacement : l'adresse, l'analyse du contenu de la page, la satisfaction a priori et le ratio entre le temps d'affichage et la taille de chaque page Web visitée ;

2. L'évaluation des pages ou des actions en termes de réussite ou échec est soit donnée par l'utilisateur soit inféré selon le point de vue du concepteur : par exemple une réservation d'une place dans un musée sera interprétée comme satisfaisant a priori l'utilisateur.

- et de deux variables de type on-trip relative à leur chemin parcouru: la *localisation* de l'utilisateur lors de sa recherche d'informations sur le web et des *informations* multivues (touristiques, géographiques) pertinentes extraites de l'analyse du contenu de la page courante à l'aide d'une ontologie dans le domaine en vue, augmentée de la localisation, de formuler automatiquement une requête au système d'information GIS utilisé.

Navigation et contexte. Ici le début et la fin d'une navigation sont indiqués soit par l'utilisateur (connexion, déconnexion) soit par un temps d'inactivité de l'utilisateur sur son navigateur Web. Le pas de temps est défini par un clic souris de l'utilisateur sur le navigateur. Le contexte comprend des informations résumées de la situation comportementale relative à sa navigation sur le Web et aussi relative à son déplacement. Pour la navigation sur le web, diverses mesures sont utilisées comme la répétition exprimant le taux de pages visitées au moins deux fois. Il comprend également par abus de langage le profil utilisateur (dispositif utilisé, centres d'intérêt, etc.) si disponible.

Représentation des cas. Cette dernière permet de sélectionner les indices comportementaux d'un cas par rapport à l'instant de référence et comprend :
- les 3 dernières pages sur les variables de l'adresse, du contenu, de la position et des informations pour identifier le contexte immédiat de réutilisation d'un cas dans une navigation (notion de restriction),
- un ensemble de pages passées qui sont sélectionnées pour leur pertinence: soit pour leur satisfaction soit pour le ratio d'affichage dépassant un seuil donné,
- et un ensemble de contraintes temporelles permettant d'ordonner les pages sélectionnées.
La partie solution d'un cas est définie par une liste d'actions à recommander par rapport à l'instant de référence, par exemple actions de sélection de telles destinations, etc.

La Figure 4 donne l'exemple d'un cas créé à partir de l'instanciation d'un patron de cas potentiel dans la navigation pour un instant correspondant à la page #9. La restriction (i.e. l'ensemble d'indices utilise dans un premier filtrage des navigations) permet de sélectionner les trois dernières pages (p=3) ainsi que la page #3 pour la satisfaction utilisateur et la page #5 pour le ratio d'affichage. A cette situation comportementale, la page #13 est associée et représente la seule recommandation de ce cas (par exemple un page indiquant une réservation dans un musée)..

Conception des phases de raisonnement. Il s'agit ici de la spécification des quatre phases du RàPC. Nous renvoyons le lecteur à la spécification du système Broadway-Web [8] auquel nous avons prévu :
- la prise en compte des deux variables pour les aspects localisation dans le raisonnement. Ces deux variables interviennent, outre dans la phase de recherche de cas, dans la phase de réutilisation, en particulier pour l'ordonnancement des recommandations selon des critères de proximité géographique et d'adéquation à la recherche courante en termes de contenu (en plus de l'adéquation à son profil).
- l'utilisation d'une ontologie dans le domaine et/ou du système d'information géographique pour l'étape d'adaptation.

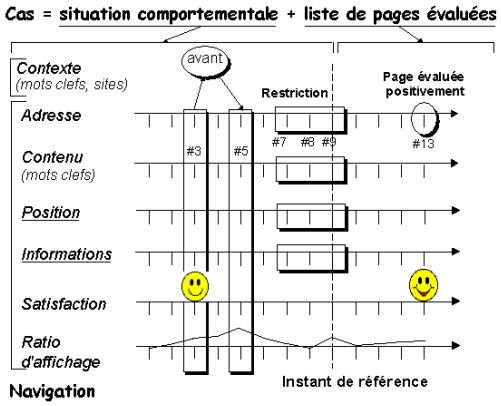


Fig. 4 – ***Exemple d'un cas issu d'une navigation***

Intégration. L'intégration d'un assistant basé sur l'approche Broadway dans un serveur d'applications mobiles consiste à définir la manière de récupérer les évènements pertinents relatifs aux actions utilisateur soient durant son interaction Homme-Machine (ici sa recherche d'information sur le web) soit durant son déplacement (ici sa localisation via GPS par exemple). Concernant l'observation de l'interaction homme-machine avec son navigateur Web, nous pouvons récupérer ces évènements soit directement via une technologie proxy (version actuelle cf. Figure 6.) soit via un serveur d'applications basées sur la localisation comme le montre la Figure 5.

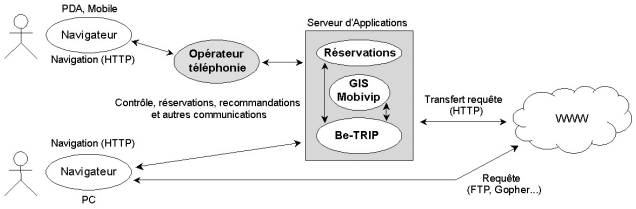


Fig. 5 – ***Intégration future de Be-TRIP dans un serveur d'applications basées sur la localisation***

Le recommandeur Be-TRIP nécessite, outre les serveurs déjà utilisés dans Broadway-web (utilisateur, analyse du contenu des pages web, annotations), un serveur d'information de type GIS (ici le futur système MobiVIP [3]). En effet afin d'expérimenter, démontrer et évaluer l'impact des nouveaux systèmes d'information sur la mobilité urbaine, l'INRIA et ses partenaires ont lancé fin 2003 le projet MobiVIP[3]. Ce projet s'inscrit dans le cadre de l'Intégration des Systèmes d'Information et de Communication du Programme National de Recherche et d'Innovation dans les Transports Terrestres - Predit.

Implémentation. Actuellement seule la version en pre-trip de Be-TRIP a été implémentée. Elle est implémenté comme un serveur HTTP utilisé en tant que proxy (cf. Figure 6) comme c'était le cas pour notre système Broadway-Web et un accès uniquement aux ressources Web via le protocole HTTP. Les requêtes effectuées dans d'autres protocoles (FTP ou Gopher par exemple) sont exécutées directement

3. MobiVIP URL: http://www-sop.inria.fr/mobivip

ou avec d'autres mécanismes indépendants de Be-TRIP.

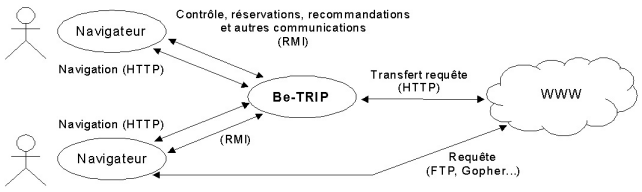


Fig. 6 – ***Architecture actuelle du système d'aide***

Le serveur de recommandation de Be-TRIP s'appuie sur notre plateforme objet en raisonnement à partir de cas appelé CBR*Tools [4] et sur notre boite à outils pour l'aide à la recherche d'information sur le Web appelée Broadway*Tools. La plate-forme objet CBR*Tools facilite le développement des systèmes implantant l'approche Broadway. En particulier CBR*Tools implémente notre *modèle d'indexation par situation comportementale*: elle comprend plus de deux cents classes Java, sa documentation basée sur la notation UML est décrite en termes de patrons de conception [8]. Elle a été conçue avec l'atelier Rose de Rational et implantée en Java.

3.3 Illustration en pre-trip

Nous avons étudié l'intérêt d'utiliser l'approche Broadway via le système Be-TRIP dans une maquette de site Web dédiée à la Communauté d'Agglomération de Sophia Antipolis (CASA) permettant ainsi de faire des liens transverses utiles pour les citoyens ou touristes. En particulier, les liens entre informations touristiques et informations pour préparer le déplacement (cartes, parcours, etc.) sont illustrées dans cette application [4].

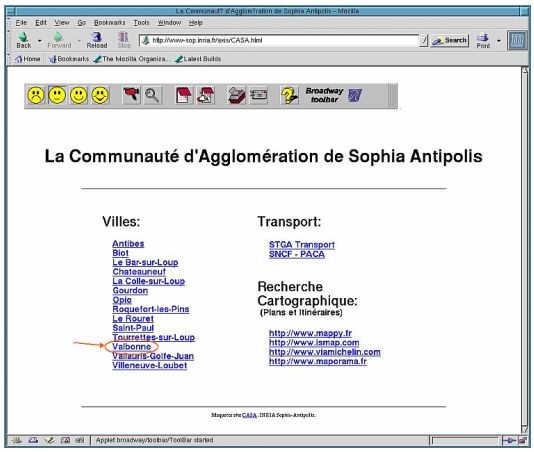


Fig. 7 – ***Session 1: sélection de Valbonne***

Pour cela, une maquette du site CASA (cf. Figure 7) a été réalisée contenant des informations sur les 14 communes qui forment la CASA. Sur les sites de chaque ville, les utilisateurs peuvent trouver des informations diverses comme des informations touristiques, des évènements particuliers, des adresses de restaurants, etc., mais les informations sur les moyens de transport ou sur les plans d'accès ne peuvent être exhaustifs. Pourtant ce type d'information est très utile pour le futur touriste, mais aussi pour tous les citoyens de ces villes cherchant un moyen de déplacement ou un plan. Dans ce but, nous avons intégré dans la maquette du site CASA pour illustrer notre approche des liens vers des sites contenant des informations sur les transports et/ou des informations cartographiques pour trouver des plans d'accès ou réaliser des itinéraires.

Nous présentons ici un scénario d'utilisation du système Be-TRIP dans le cadre du site Web de la CASA pour les touristes. Prenons l'exemple d'un touriste, séjournant dans la ville de Valbonne qui se connecte au site de la CASA (cf. Figure 7). Celui-ci commence à rechercher, en navigant dans le site, quelque chose à visiter dans l'après midi. Il trouve la célèbre abbaye fondée en 1199. Étant donné qu'il ne trouve pas l'adresse de cette abbaye, il continue à chercher et la trouve enfin après plusieurs clics souris (cf. Figure 8).

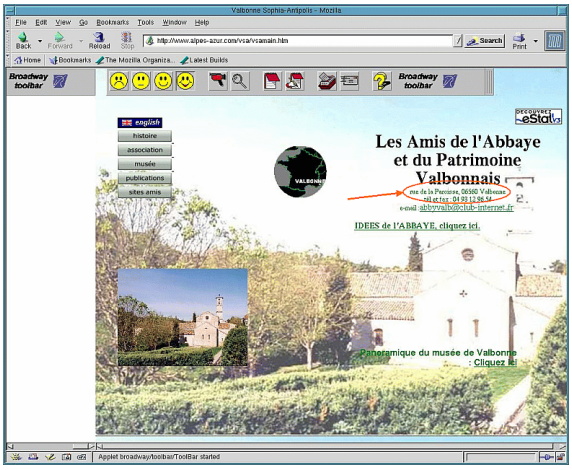


Fig. 8 – ***Session 1: adresse de l'abbaye trouvée***

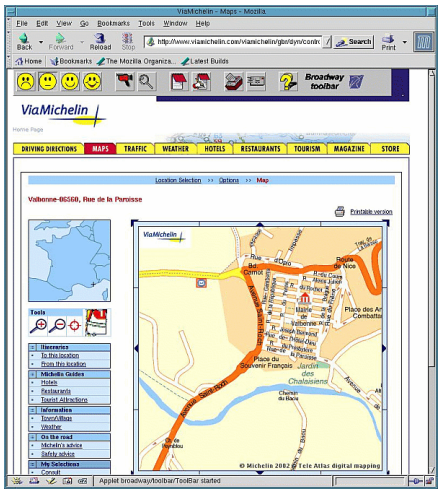


Fig. 9 – ***Session 1 (fin): sélection de la carte suite à une requête sur viamichelin***

Après avoir noté l'adresse, il retourne sur la page d'accueil du site pour se rendre ensuite sur le site de www.viamichelin.fr afin d'avoir l'itinéraire. Après avoir effectué sa requête, il obtient le plan d'accès (cf. Figure 9) et se déconnecte donc

4. URL http://www-sop.inria.fr/axis/cbrtools/manual/

satisfait.

Imaginons maintenant un autre touriste naviguant dans le site de la CASA de façon similaire, en particulier sur les pages de l'abbaye de Valbonne. Be-TRIP lui propose sans qu'il fasse la requête sur www.viamichelin.com plusieurs recommandations dont le plan d'accès trouvé par le premier touriste (cf. Figure 10).

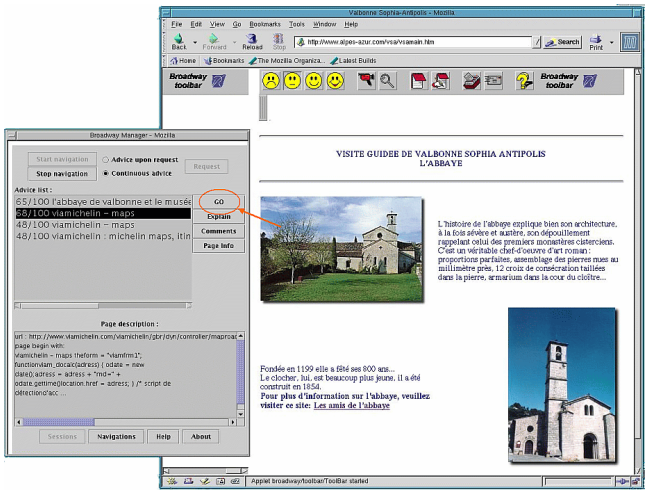


Fig. 10 – ***Session 2: aide personnalisée par similarité de comportements utilisateur***

4. COMPARAISON AVEC DES TRAVAUX SIMILAIRES

Cette section décrit brièvement quelques travaux proches de nos recherches relatives à la personnalisation de services d'aide à la mobilité puis en présente une synthèse dans le tableau 1.

Le projet IMAGE [3] propose à ses utilisateurs (via des PC, PDA et mobiles) des informations personnalisées basées sur leur position (LBS) et sur des techniques de "user profiling". Les citoyens et les touristes sont les utilisateurs du système qui offre aussi la possibilité de "e-ticketing" ou de téléchargement des informations nécessaires.
Le but du projet WH@M [7] est d'aider les touristes dans la phase de pre-trip (calcul d'itinéraires) et de leur fournir des informations en temps réel (météo, trafic, etc.) pendant leur voyage. L'information est adaptée en fonction de la location et du profil de l'utilisateur qui peut aussi bénéficier du "feed-back" des autres utilisateurs (forums, opinions sur des "items").
Dans le cadre du projet CRUMPET [9], la personnalisation est basse sur les trois facteurs suivantes : la localisation, l'adaptation aux intérêts et préférences de l'utilisateur, et l'adaptabilité du profil de l'utilisateur à partir de l'historique de ses interactions ("implicit feedback"). Les services proposées visent les utilisateurs mobiles comme les touristes utilisant des réseaux mobiles (PDA, mobiles, etc.) ou fixes ("travel-kiosks").
Le système de recommandations développé dans le cadre du projet DIETORECS [6] est un système interactive (questions-réponses) qui aide le future touriste (pre-trip) à sélectionner sa destination, sous la forme d'un "TravelPlan". Des techniques de Raisonnement à Partir de Cas (RàPC) sont employée pour l'aide à la décision. Le système AVC (Augmented Virtual City) du projet PALIO [2] adapte l'information fournie aux touristes et citoyens en tenant compte de la technologie utilisée, des caractéristiques du réseau, de la position de l'utilisateur, de son profil, du contexte actuel et aussi de l'historique de ses interactions avec le système.

Nous décrivons ces systèmes selon plusieurs caractéristiques: a) le contexte de l'aide (pre ou/et on-trip), b) le mode de recherche (par requête ou par "browsing") visé par l'aide, c) l'origine des données utilisées pour calculer l'aide personnalisée, d) l'approche de recommandation utilisée (A.P.U ou/et F.C.), e) les modes pour fournir l'aide (push ou pull ou les deux) et enfin f) l'utilisation ou non d'une corrélation entre les deux types de comportements utilisateurs (séquence d'actions lors de sa recherche d'informations et déplacement).

On constate que la plupart des travaux actuels visant une aide personnalisée au déplacement à la fois en pre-trip et/ou on-trip s'appuient essentiellement sur l'apprentissage du profil utilisateur sans utiliser des données issues d'autres utilisateurs. A l'inverse, citons néanmoins trois travaux offrant un filtrage collaboratif : Wh@m qui utilise des annotations issues d'autres utilisateurs et les deux travaux basés sur des techniques RàPC (Dietorecs et Be-TRIP) qui réutilisent des cas i.e des expériences issues de sessions utilisateurs.

A notre connaissance, l'originalité principale de Be-TRIP réside dans les quatre points suivants:
a) Be-TRIP propose des recommandations personnalisées suite à un processus de recherche d'informations par *browsing* alors que les autres travaux se placent tous dans une recherche par formulation de requête.
b) Be-TRIP due à l'approche Broawday utilisée est le seul à prendre en compte l'ordre des actions utilisateur dans sa recherche d'information mais aussi son déplacement.
c) Be-TRIP propose un calcul de recommandation basé sur des similarités de comportements utilisateur en recherche d'information sur le Web corrélés à son déplacement. Certains travaux de personnalisation s'appuie sur un seul historique : citons Palio (recherche d'informations), Crumpet (déplacement).
d) Enfin Be-TRIP grâce au raisonnement à partir de cas a des capacités d'apprentissage en-ligne de nouveaux cas comme dans Dietorecs mais aussi sur la qualité des recommandations proposées: en effet l'utilisateur étant dans la boucle du raisonnement (dans la phase de révision), le système apprend des hypothèses de recommandations qui s'avèreraient fausses i.e non choisies par l'utilisateur.

5. CONCLUSIONS ET PERSPECTIVES

Pour conclure, nous pensons qu'associée à la politique générale des déplacements d'une communauté d'agglomération par exemple, les systèmes de recommandation sont de nouveaux outils au service de stratégies socio-économiques, environnementales, touristiques et culturelles (itinéraires thématiques) associant à la fois contenu et optimisation des déplacements, en particulier les transports en commun.

Dans cet article, nous avons spécifié Be-TRIP un assistant au déplacement pre-trip et on-trip basé sur une approche

Auteurs	Aide	Recherche par	Données	Approche	Modes	Corrélation recherche/déplacement
Image [3]	on-trip	requête	mono-utilisateur	A.P.U	pull	non
Wh@m [7]	pré/on-trip	requête	groupe	A.P.U + F.C.	pull	non
Crumpet [9]	on-trip	requête	mono utilisateur	A.P.U	push/pull	non
Dietorecs [6]	pre-trip	requête	groupe	A.P.U et F.C. (RàPC)	pull	non
Palio [2]	pré/on-trip	requête	mono utilisateur	A.P.U	pull	non
Be-TRIP	pré/on-trip	browsing	groupe	A.P.U. et F.C. (RàPC)	push/pull	oui

Tab. 1 – *Tableau récapitulatif*

originale de calcul de recommandations personnalisées, l'approche Broawday. L'originalité de cet assistant réside essentiellement dans l'utilisation de deux niveaux de personnalisation corrélés aussi bien en pre-trip qu'en on-trip: a) un basé sur l'historique de déplacement de l'utilisateur et le contexte géographique dans lequel il se trouve lors de la recherche d'information et b) un relatif à l'historique de sa recherche sur Internet en termes de contenu, etc.

Les travaux en cours et futurs à relier avec le projet Mobi-VIP concernent plusieurs aspects:
- La poursuite de la spécification et la réalisation de la prise en compte des aspects localisation et dispositif de l'utilisateur en déplacement et l'utilisation de connaissances dans le domaine visé dans l'étape d'adaptation de cas; son intégration dans le serveur d'applications visé;
- L'évaluation de l'aide apportée par notre maquette auprès d'utilisateurs via des expérimentations;
- L'étude de l'application de recherches menées dans l'équipe (soutenues par la région Paca) en classification hiérarchique ascendante [5] pour la classification de profils de comportements utilisateur corrélant divers contextes d'utilisation différents: par exemple selon la localisation, selon le type de comportement (recherche ou déplacement réel) ou selon les dispositifs utilisés;
- Et enfin l'analyse de la qualité des recommandations pour l'aide à la reconception du système d'information.

6. REFERENCES

[1] A. Aamodt and E. Plaza. Case-Based Reasoning : Foundational Issues, Methodological Va riations, and System Approaches. *The European Journal oo Artificial Intelligence*, 7(1):39–59, 1994.

[2] A. Andreadis, G. Benelli, A. Bianchi, P. Fedele, and G. Giambene. Universal Access to Personalised Information Services. Technical report, Dipartimento di Ingegneria dell'Informazione, Universita di Siena, Siena, Italy, July 2001. IST-PALIO project.

[3] P. Blythe, S. Edwards, and S. Scott. Delivering dynamic, personalised travel and tourist information through mobile phones and PDAs: the IMAGE project. In *10th World Congress and Exhibition on INTELLIGENT TRANSPORT SYSTEMS AND SERVICES*, 16 - 20 November 2003.

[4] S. Chelcea. Etude, implémentation et évaluation d'un algorithme de classification pour des systémes de recommandations dans le domaine du tourisme. UNSA Internship report, juillet 2002. Sophia Antipolis, France.

[5] S. Chelcea, P. Bertrand, and B. Trousse. Un Nouvel Algorithme de Classification Ascendante 2-3 Hiérarchique. In *Reconnaissance des Formes et d'Intelligence Artificielle (RFIA 2004)*, Centre de Congrès Pierre Baudis, Toulouse, France, 28-30 Janvier 2004.

[6] D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber, and C. Zanellai. DIETORECS: Travel Advisory for Multiple Decision Styles. In *Enter conference*, Helsinki, Finland, January 29 - 31 2003.

[7] IST project for tourism. WH@M: The World in your H@nds on the Move. Congress on IST - ITS, 20-23 June 2001. Bilbao, Spain.

[8] M. Jaczynski. *Modèle et plate-forme à objets pour l'indexation des cas par situation comportementale: application à l'assistance à la navigation sur le Web.* PhD thesis, Université de Nice-Sophia Antipolis, Sophia-Antipolis, France, décembre 1998.

[9] S. Posla, H. Laamanen, R. Malaka, A. Nick, P. B. e, and A. Zipf. CRUMPET: Creation of User-friendly Mobile Services Personalise d for Tourism . In *IEE 3G 2001 conference*, pages 26–28, London, UK, March 2001.

[10] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[11] B. Trousse, M. Jaczynski, and R. Kanawati. Une approche fondée sur le raisonnement à partir de cas pour l'aide à la navigation dans un hypermédia. In J.-P. Balpe, S. Natkin, A. Lelu, and I. Saleh, editors, *Proceedings of Hypertexte & Hypermedia : Products, Tools and Methods (H2PTM'99)*, pages 13–26. hermes, august 1999. Paris.

[12] A. Wexelblat and P. Maes. Footprints: Visualizing histories for web browsing. In *Proceedings of RIAO'97, Computer Assisted Informat ion Retrieval on the Internet*, Montreal, Canada, 1997.

[13] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linki ng. *Computer Network and ISDN systems*, 28:1007–1014, mai 1996. (proceedings of the 5th international WWW conference).